Universidade de Évora - Instituto de Investigação e Formação Avançada

Programa de Doutoramento em Informática

Tese de Doutoramento

Mapping in uncertain environments for mobile robots

Hongjun Li

Orientador(es) | Miguel Barão
Luis Rato

Évora 2019

Universidade de Évora - Instituto de Investigação e Formação Avançada

Programa de Doutoramento em Informática

Tese de Doutoramento

Mapping in uncertain environments for mobile robots

Hongjun Li

Orientador(es) | Miguel Barão
Luis Rato

Évora 2019

A tese de doutoramento foi objeto de apreciação e discussão pública pelo seguinte júri nomeado pelo Diretor da Instituto de Investigação e Formação Avançada:

- Presidente | Paulo Quaresma (Universidade de Évora)
- Vogal | Luís Paulo Gonçalves dos Reis (Universidade do Porto)
- Vogal | Luís Miguel Parreira e Correia (Universidade de Lisboa - Faculdade de Ciências)
- Vogal | Rodrigo Ventura (Instituto Superior Técnico)
- Vogal | Pedro Salgueiro (Universidade de Évora)
- Vogal-orientador | Miguel Barão (Universidade de Évora)

Évora 2019

To my dearly beloved parents Huaimin Li and Wenzhi Tao, and my precious sister Chunhong Li.

Acknowledgments

First and foremost, I would like to thank my supervisors Professor Miguel Barão and Professor Luís Rato for their patience and guidance, and for spending a lot of time sharing their knowledge. Without their help, this work would not have been successful.

This work was supported by EACEA under the Erasmus Mundus Action 2, Strand 1 project LEADER - Links in Europe and Asia for engineering, eDucation, Enterprise and Research exchanges. I also would like to thank the coordinator Professor Teresa Gonçalves who help me a lot during my time here.

I would like to thank all the PhD students and Master students in our lab for their help and friendship. Thanks also to many people who have given me help during my PhD.

Finally, I am grateful to my parents and my sister for their love and support.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

BOF Bayesian Occupancy Filtering

CCD Charge Coupled Device

DATMO Detection and Tracking of Moving Objects

EDHMM Explicit-state-Duration HMM

EKF Extended Kalman Filter

EM Expectation Maximization

GPOM Gaussian Process Occupancy Map

GRF Gaussian Random Field

HMM Hidden Markov Model

ICP Iterative Closest Point

IDC Iterative Dual Correspondence

IMRP Iterative Matching Range Point

IOHMM Input-Output HMM

IR Infrared

MRF Markov Fandom Field

P-SLAM Prediction-based SLAM

RBF Radial Basis Function

ROMA Robot Object Mapping Algorithm

SIFT Scale Invariant Feature Transform

SLAM Simultaneous Localization And Mapping

SMC Sequential Monte Carlo

UKF Unscented Kalman Filter

Sumário

Mapeamento em ambientes incertos para robôs móveis

Um dos problemas fundamentais em robótica móvel é o problema da localização e mapeamento, no qual um robô se deve localizar ao mesmo tempo que constrói um mapa do ambiente. Existem diversas técnicas para abordar este problema. Neste trabalho propõem-se abordagens novas para a construção do mapa em ambientes estáticos e dinâmicos, assumindo pose conhecida.

As abordagens propostas baseiam-se em campos aleatórios de Markov (Markov random fields - MRF) e em campos aleatórios Gaussianos (Gaussian random fields - GRF), seguindo um ponto de vista Bayesiano, onde as distribuições de probabilidade a priori são usadas como regularizadores. Num ambiente estático, cada ponto do espaço é descrito pela sua probabilidade de ocupação. O primeiro método proposto é um filtro baseado nos MRF, que se centra no ruído das medidas e que pode ser implementado em linha (tempo real). O segundo método é um método preditivo baseado nos MRF que permite também estimar a probabilidade de ocupação do espaço não observado. Em ambos os métodos, os mapas são construídos numa grelha de células. Outra abordagem baseia-se num espaço contínuo, baseado em GRF onde se propõe um método recursivo de modo a reduzir a complexidade computacional.

No caso de ambientes dinâmicos, a probabilidade de ocupação é substituída pelas probabilidade de transição duma cadeia de Markov para descrever o comportamento dinâmico de cada ponto. Nesta abordagem são propostos dois métodos para os ambientes dinâmicos, igualmente baseados nos MRF e nos GRF. No método com MRF todos os parâmetros são estimados em conjunto. Pelo contrário, com os GRF os parâmetros são divididos em dois sub-conjuntos de modo a reduzir a complexidade computacional.

Todos os métodos propostos são testados e apresentam-se resultados em simulação nos respetivos capítulos. Finalmente estes algoritmos são também validados em ambiente experimental. Nestas experiências, as poses não podem ser medidas com precisão e é tida em consideração a incerteza na pose do robô.

Quando comparados com o estado da arte, os métodos propostos resolvem as inconsistências nos mapas tendo em consideração a dependência entre pontos vizinhos. Este processo é realizado usando MRF e GRF em vez de assumir independência. As simulações e os resultados experimentais demonstram que os métodos propostos podem, não apenas lidar com as inconsistências nos mapas construídos, mas também tirar proveito da correlação espacial para prever o espaço não observado.

Palavras chave: Campos aleatórios de Markov, Campos aleatórios Gaussianos, Robótica móvel, Mapeamento e localização

Abstract

Mapping in Uncertain Environments for Mobile Robots

One of the fundamental problems in robotics is the localization and mapping problem, where a robot has to localize itself while building a map of the environment. Several techniques exist to tackle this problem. This work proposes novel mapping approaches with known robot poses for static and dynamic environments.

The proposed techniques are based on Markov random fields (MRFs) and Gaussian random fields (GRFs), following a Bayesian viewpoint where prior distributions are provided as regularizers. In static environments, every point is described by its occupancy probability. The first proposed method is an MRF-based filter, which focuses on the measurement noise and can be implemented online (realtime). The second one is an MRF-based prediction method, which can also be used to estimate the occupancy probability for unobserved space. In both methods, the maps are organized as a grid. Another approach, which works in continuous space, is based on a GRF prediction method, and a recursive algorithm is proposed to reduce the computational complexity.

In the case of dynamic environments, the occupancy probability is replaced by transition probabilities of a Markov chain that describe the dynamic behaviour of each point. Two methods for dynamic environments are proposed, also based on MRFs and GRFs. In the MRF-based method, all the parameters are jointly estimated. In contrast, in the GRF-based method, the parameters are divided into two subsets to reduce the computational complexity.

All the proposed methods are tested in simulations in the corresponding chapters. Finally, these algorithms are also validated on an experimental platform. In the experimental environments, robot poses cannot be measured precisely, and so the uncertainty of robot poses is also considered.

When compared with the state of the art for dynamic environments, the proposed methods tackle the inconsistencies in the maps by considering dependence between neighbour points. This is done using MRFs and GRFs instead of assuming independence. The simulations and the experimental results demonstrate that the proposed methods can, not only deal with the inconsistency in the built maps, but also take advantage of the spatial correlation to predict unobserved space.

Keywords: Markov random field, Gaussian random field, Mobile robotics, Mapping and localization

1

Introduction

Since the industrial revolution in the XVIII and XIX centuries, machines have helped humans in many physical activities, improving efficiency in characteristics like speed, power, safety, precision and reliability. Robotics is one more step in the technological evolution where machines acquire the capability to operate by themselves without human intervention.

For decades, robots have been mostly used for limited and repetitive tasks that could be programmed a priori. These robots required known environments and were used mostly in industrial applications to replace human labour and increase productivity. Later, a surge in mobile robotics with applications in land, underwater, aerial and space exploration led to the need of dealing with the unknown. Robots became autonomous, which required information from the environment and taking decisions without human intervention.

In more recent times, autonomous robots are entering our everyday life. Robotic vacuum cleaners and autonomous driving cars are two representative examples, but many more examples are emerging in our society.

Dealing with uncertainty is one of the challenges that must be addressed in mobile robotics. When an autonomous robot explores unknown environments, it should have the ability to learn the environment.

After perceiving the environment by sensors, the robot should be able to build a map, which is useful for itself to run into the environment again and do further tasks. Normally, observations are obtained in local coordinates depending on robot poses. When building a map, observations and robot poses must be located in the same coordinates. Robot poses are based on its kinematic model, and they are also noisy. The poses can also be corrected when the robot senses the same place more than once. Given observations, the robot should build a map and localize itself at the same time which is the fundamental problem called simultaneous localization and mapping (SLAM) [LDW91]. It can be factored into a localization problem and a conditional mapping problem [DWB06]. The objective of the localization problem is to estimate robot poses, and the mapping problem is to build maps. The SLAM problem has been studied by many researchers, and there are various methods to solve it, such as Extended Kalman filter (EKF) SLAM and Fast-SLAM [MTKW02].

All the information is obtained from sensors. Range finders, such as laser sensors and sonar sensors, can only detect their distances to the surrounding objects. Cameras, on the other hand, can capture more information but can not directly measure distance. The depth of objects to cameras can be estimated by feature or optical flow based methods [HN06] [BH09]. Moreover, cameras can also be used to track dynamic objects, such as people [PKK93]. Compared with range finders, cameras require higher processing power.

1.1 Motivation

Indoor service robots are becoming very popular now. Normally dynamic objects in indoor environments, such as doors and pieces of furniture, move slowly. In low dynamic environments, robots only need to know where to go and need not to track objects. Because of the cheap price and low processing power requirement, range finders are normally equipped onboard indoor service robots, such as robotic vacuum cleaners. Because of the uncertainty of range sensors, the built maps are noisy. This thesis focuses on how to make full use of the information from range sensors to build smooth maps that help navigate indoor robots.

Both in static and dynamic environments, one point is similar to its surroundings in some properties. For example, one point in a static environment is more likely to be occupied when the points near it are observed occupied. If it is observed free, its observation is more likely to be noisy, and its observed state may be corrected. Even if it is not observed, there can be a guess that it is occupied. This spatial correlation can be modelled by Markov random fields (MRFs) or Gaussian random fields (GRFs). MRFs are widely used in image analysis [Li09]. Meanwhile, the built maps can be represented by images. MRFs can be applied to deal with the inconsistence in maps [TTW+04]. GRFs describe the correlation in continuous space. Given some points in a GRF, any point in continuous space can be predicted [OR12]. Assuming each point is correlated to its surrounding points, the built map for observed space will be smoother. Although there is no data from unobserved space, the map can be predicted from surrounding observed space.

For static environments, some researches, such as [OR12] and [TTW+04], apply GRFs and MRFs in robot mapping. However, their computational complexities are very high. In static environments, normally one point is assumed to have two states: occupied and free [Elf89]. When points are represented by discrete values, the curse of dimensionality will make many algorithms infeasible in practice. In order to reduce the computational complexity, a better representation for static environments is occupancy probability or free probability. In dynamic environments, robots have to interact with unpredictable and dynamic objects. When focusing on one point, its state can change with time, so a Markov chain can be used to model the dynamic behaviour for every point [MDBB12]. The parameters of Markov chains are used to represent dynamic environments. Normally the correlation between parameters in space is not considered. In this

thesis, the GRFs and MRFs are also applied in dynamic environments to smooth and predict dynamic environments.

1.2 Objectives

The main objectives of this thesis are to solve the mapping problem in slow dynamic environments, where the dynamics come from obstacles that can appear and disappear in time. The map representation should have information about the dynamical behaviour of every point in space and enable other tasks (like path planning) to be performed while taking this information into account.

While there are already methods in the literature to tackle this problem, the solutions proposed there assume that each point in space is independent of the others, which is not realistic. A more realistic approach would be to assume that if a point is dynamic, its immediate neighbourhood also has similar dynamics with high probability. The aim of this thesis is to consider the correlated nature of the dynamics between different points in space to improve the maps obtained. As expected benefits, the resulting maps would be smoother, and it would also allow prediction of unobserved regions to be performed, for example to interpolate two nearby observations.

1.3 Main contributions

This thesis proposes several methods based on MRFs and GRFs to build maps for static and dynamic environments. The proposed methods use correlations between nearby points in space to filter observations and to predict unobserved space.

The main contributions of the thesis are the following:

- A new MRF-based filter for static environments using the log odds form. The objective is to filter out the noise in observations and can be implemented online. The main advantage of using the log odds form is the existence of a feasible solution when considering spatial correlation in occupancy states [LBaR].

- A new MRF-based filtering and prediction for static environments using the log odds form. This method predicts the occupancy probabilities of observed and unobserved space, and the algorithm can be computed recursively. This method is an extension of the filter problem above [LBaR18c].

- A new GRF-based filtering and prediction algorithm for static environments using the log odds form. This approach also predicts the occupancy probabilities of observed and unobserved space. It can be used to build local high-resolution maps. Instead of the well-known prediction equation in Gaussian processes, a new recursive algorithm for Gaussian processes with independent noise in observations is proposed. It avoids inverting matrices and puts the algorithm in a lower computational complexity class than the previous GRF-based method [LBaR18a].

- A new MRF-based filtering and prediction algorithm for dynamic environments. Markov chains are applied to model dynamic environments. This approach allows predicting transition matrices of observed and unobserved space [LBaR18b].

- A new GRF-based filtering and prediction algorithm for dynamic environments. It is also based on Markov chains and predicts the transition matrices of observed and unobserved space.

1.4 Thesis outline

In Chapter 2, a brief overview of mapping is presented, and the state-of-the-art methods related to range finders, occupancy grid mapping, statistic processes in mapping, are reviewed.

In Chapter 3, the fundamental materials that will be used throughout this thesis are introduced. The basic probabilistic theories are described. In addition, some statistic processes, such as Gaussian process and Markov chain, are introduced. Finally, unscented Kalman filters are presented.

In Chapter 4, occupancy probabilities are applied to represent static environments, and several approaches are proposed. Based on MRFs, a filter is presented focusing on the noise in observed space, and a prediction is proposed to predict the occupancy probabilities of observed and unobserved space. These two methods build maps in discrete space. In addition, a GRF-based prediction is described, which works in continuous space. A recursive method is proposed to avoid the inverse matrix problem in Gaussian processes. These methods are tested in simulations.

In Chapter 5, Markov chains are applied to model dynamic environments, and two approaches are proposed based on MRFs and GRFs, respectively. These two approaches can predict the transition matrices of observed and unobserved space. The GRF-based method is divided into two steps to reduce computational complexity. These methods are tested in simulations.

In Chapter 6, the previously proposed approaches are evaluated on a 3pi robot-based experimental platform. A 3pi robot equipped with two infrared (IR) sensors is used to perceive the environment. In experimental environments, the uncertainty of robot poses is also considered. Based on the experimental data, these approaches are tested incorporating the uncertainty of robot poses.

In Chapter 7, conclusions of this thesis and future work are provided.

2

State of the art

2.1 A brief overview of mapping

Robot mapping is the task of creating maps when robots explore unknown environments. With these maps, robots can do further tasks, such as navigation and path planning. There are three representative kinds of maps: metric maps, topological maps, and semantic maps [DDK15]. The most common metric maps are geometric maps, feature maps and grid maps. In the geometric maps, objects are represented by their shapes. Only the objects are stored, and less memory is required. In the feature maps, environments are represented by features extracted from the scans of the sensors. Features could be geometric features or visual features, which depend on the sensors available. For sonar sensors [ATTM07], the features are jump-edges and corners detected in the scans. For visual sensors, features are extracted from images, and usually are based on the scale invariant feature transform (SIFT) [Low99]. A grid map represents the environment by many grid cells, and the most typical grid map is the occupancy grid maps proposed in [Elf89], where occupancy probabilities are used to describe grid cells. Many details of environments are stored in grid maps which are convenient for exploration. However, the maps require a lot of memory to store these grid cells. The topological maps [Cho96], on the other hand, have an approach that represents the environment in a much more compact way. This approach models space structures as graphs where the

objects or places are represented by nodes, and arcs are the paths and distances between these nodes. The semantic maps [KB91] [KMEM11] store high-level information. Not only objects but also their properties are provided.

In early research [Elf86], maps were mainly for robot navigation. Relative spatial relationships are considered in [SSC90] where the estimates of landmarks are correlated with each other because of the uncertainty in robot poses. The mapping and localization are considered as the "chicken and egg" problem in [LDW91], where the simultaneous localization and mapping (SLAM) is initially proposed.

Extended Kalman filter (EKF) SLAM [DNDW+00] is a remarkable solution to the SLAM problem under a Gaussian noise assumption. Since robot motion and observation models are not linear, an EKF is applied to linearize them. However, if these models are highly non-linear, the error is intolerable. Another valuable solution is Fast-SLAM [MTKW02], which applies a particle filter to deal with the pose uncertainty. In Fast-SLAM, particles are sampled from a *proposal distribution*. The weighted particles can be used to approximate the true pose distribution, and the weights can be updated recursively. EKFs are used for different landmarks individually. This work is extended in [MSDB03]. The *proposal distribution* is the pose conditioned on not only inputs and the past poses but also observations. For each particle, its weight has the smallest variance given the conditions.

Loop closure is a task of deciding whether or not a robot returns to a previously visited place. Normally, robot positions are estimated based on the motion model. Because of the inherent noise in the robot, the uncertainty of robot positions will become bigger and bigger. As a result, the robot cannot close its trajectory when it returns to the previously visited place. If the robot recognizes this place, it can help to reduce the uncertainty of the robot positions and mapping results. In order to solve this problem, the robot should label special places, which are features extracted from the observations. With more distinguishing features, the robot has more chance to close its trajectory. A range sensor is used in [GSNR11] to close the robot trajectory. The features, which contain the important geometric and statistical information, are extracted from the point clouds. A booting classifier is trained to compare two features and decide whether they are the same. For a 3D lidar, histogram based signatures are extracted as features [ML11]. Each new histogram is compared with previously stored histograms. If the distance between two histograms is less than a threshold, the trajectory can be closed. The iterative closest point (ICP) algorithm [BM92] is applied to compute the similarity of two surfaces [Low04]. For every point in one surface, the ICP will give the minimum distance to the other surface. The similarity is then the sum of squared distances. For the loop closure based on images, the similarity is based on the SIFT descriptor [NCH06] [HN06]. The multi-level surface map is proposed for outdoor environments in [TPB06]. Multiple surfaces are stored in each grid cell, and the robot can close its trajectory even when there are crossing bridges or underpasses. The loop closure is easier for topological maps with planar constraints. If there is a crossing edge, the cross will be a loop closure [SK04]. Without enough observations, it is difficult to detect a loop closure. An active loop closure is proposed in [SHBG05], where if there is an opportunity for a loop closure, the robot can move further and obtain more information to check if there is a loop closure.

2.2 Range finders for robotics

Range finders measure distances from sensors to targets. The commonly used range finders are sonars, laser sensors, radars, and IR sensors. Because of their cheap cost and relatively accurate ranges, sonars were used for robot mapping and navigation in early research [ME85]. For indoor environments, the target types, such as planes, corners, and edges, can be extracted for accurate mapping [KK95]. The target classification is done online in [HK01]. Sonars can also be used in underwater environments [SNKB07] [RRN10]. Sonars also have disadvantages, such as angular uncertainty, specular reflections, and crosstalk. Lasers offer precise

measurements and are now popular for robotics. For indoor environments, the lines and corners can also be extracted for scan matching [ATTM07]. 3D maps are built using lasers for indoor [GNB00] and outdoor [CN06] environments. In addition, the use of laser sensors in a smoky environment [FDvNV10] and a glass-walled environment[KC16] are also explored. In [DK04], sonar and laser data are fused for SLAM. The main disadvantage of laser sensors is their expensive cost. IR sensors are very cheap and have high refresh rates. However, they are sensitive to surface reflectance properties. Because of that, they are used for obstacle avoidance in robotics. IR sensors have been applied in multi-robot systems [MS04] in recent years where the neighbouring robots can measure the orientations and distances of their neighbours. With high refresh rates, the robot can detect high dynamic objects and avoid collisions. A 2.5D IR range and bearing system [RSZF09] and a new prototype dual rotating IR sensors [LC11] are designed for muti-robot systems. In [GLGMM+11], an IR sensor system is used to estimate the position of robots in an intelligent space [LH02], where what inhabitants are doing can be understood through distributed sensors. Since the radar works well in bad weather conditions, it is suitable to overcome problems for assistance functions relying solely on camera data [WLH+17]. A radar is used to detect road parked vehicles in [DHS+14]. 3D localization and mapping based only on the radar is proposed in [HR16]. Lasers can often lead to ghost movements for big, contiguous, stationary obstacles, a radar is used to reduce these ghost movements in [NYK+15].

3D vision is another common way of building depth maps. While not strictly a range finder sensor, the use of multiple images enables 3D depth perception and the 3-dimensional reconstruction of the environment to be performed. The depth of surrounding environments can be estimated by matching features [HN06] or pixels [BH09] in multiple images. As a result, texture richness is required. For a single camera [DRMS07], multiple images in different time instants are used for depth estimation. Multiple images are obtained simultaneously for multiple cameras, such as binocular cameras [WL10] and stereo vision [HKD07]. Compared with range finders, 3D vision requires higher processing power for depth estimation.

2.3 Occupancy mapping in static environments

2.3.1 Occupancy grid mapping

In classical occupancy grid mapping, the states of different grid cells are independent of each other [Elf89]. Each grid cell can be updated recursively based on the Bayes rule when a new observation is obtained. An inverse sensor model, which is a posterior distribution of one grid cell conditional on observations, can make the computation efficient. An approximate function of the inverse sensor model is trained in [TBF05]. The data is sampled from a random map, poses, and measurements. The learning computation is complex and the learning result depends on the learning algorithm and the samples used. An exact solution to the inverse sensor model is given in [KLAM16], which is computed efficiently within the area that sensors cover. The inverse sensor model is extended to agriculture applications in [KKCR18]. Dual inverse sensor model for radar is proposed to avoid false detections [SD19]. A different approach based on a forward sensor model is used to do the mapping problem in [Thr01]. The expectation maximization (EM) is used to maximize the data likelihood and obtain the best estimation of the map. Using a Laplacian approximation, a probabilistic map can then be obtained. A novel "sensor cause model" is proposed to consider the dependence between voxels in a 3D measurement cone [AMHHS17]. Not only the occupancy probability but also the corresponding variance is provided, which is crucial for planning over grid maps.

Maps can also be improved by matching approaches. The well-known point to point matching approaches are the iterative closest point, the iterative matching range point (IMRP)[LM97], and the iterative dual correspondence (IDC) [LM97]. A "weighted" matching algorithm is proposed in [PKRB02]. Each scan point is assigned a weight based on its uncertainty. In [BS03], each cell is assigned a normal distribution

to model the probability of measuring a point locally. The matching problem is differentiable and can be solved using Newton's algorithm. The translation and orientation of the sensor are considered at the same time in [MLM05]. Feature-based scan match methods can build more accurate maps. The features for range finders are lines and corners [ATTM07] [ATT08]. In [KTR+17], maps are labelled as binary images. The graphs extracted from the images are used to estimate a similarity transformation matrix by graph matching. The similarity transformation matrix can then be used to align occupancy grid maps. Sometimes scan match can only ensure local consistency and the global map is inconsistent, Markov random fields (MRFs) are applied to smooth the global map [TTW+04]. However, the optimal solution can not be obtained in closed form.

If map sizes increase or their resolutions become high, a lot of memory is required. In order to solve this problem, mapping approaches with different resolutions are proposed by some researchers. Probabilistic quadtrees are developed for variable-resolution mapping of probabilistic occupancy data [KGU04]. The occupancy values are divided into several classes, and neighbouring cells with the same class are merged. Adaptive resolution based on measurements is proposed in [ESG10]. Neighbouring cells with similar occupancy values are merged to reduce the amount of memory required. If new measurements conflicts with the current map estimate, the affected cells are split into smaller cells. The adaption can be done online. In [JKK16], the cells along free space borders are refined with smaller sizes. The sparse 3D scans are registered with different resolutions in [DSB17]. The scans near the robot have higher resolutions and decrease with increasing distance.

2.3.2 Continuous occupancy mapping

The Gaussian process occupancy map (GPOM) [OR12], also known as Gaussian random field, is an occupancy representation of static environments in continuous space. A Gaussian process is applied to predict unknown space. The training data is made of points sampled from measurements. These points have binary values: +1 or -1 for occupied and free states. The occupancy mapping is considered as a binary classification problem. The high computational complexity of an inverse covariance matrix is the key disadvantage of the Gaussian process. For large-scale environments, a mixture of Gaussian processes using Bayesian Committee Machines [Tre00] is proposed in [KK12]. The training data is divided into many clusters, and a Gaussian process is applied to each subset. Finally, a mixture of Gaussian processes is proposed to merge these submaps into one global map. This work is extended to update the Gaussian process recursively in [KK11]. Gaussian mixture models are applied to build occupancy maps with variable resolution in [OTM18]. In order to ensure continuity, local Gaussian processes are applied to overlap clusters [KK13]. A Multi-support Gaussian process is developed to reduce the size of input data [VR16]. The map is divided into many free, occupied, or ambiguous areas. These areas are the inputs of the Gaussian process, and the multi-support kernel can deal with the two-dimensional regions. An online continuous mapping is proposed to build a map as the zero level set of a Gaussian process implicit surface [LZHL19]. A nested Bayesian committee machine is proposed to fuse the mapping result from new observations to the existing global occupancy map [WE16]. As a result, new observations can be processed locally using Gaussian process, and the mapping can be done in real time.

Another kind of continuous occupancy map is proposed in [RO16]. The map prediction is regarded as a logistic regression classification, where data is projected into a Hilbert space to reduce the dimension and complexity that arises with increasing data. A large number of features are computed from the training data, and their dot product can approximate the radial basis function kernel (RBF) [SS01]. The objective function of the logistic regression model is convex in the parameters, and the logistic regression model can be trained online by stochastic gradient descent. An overlapping method is proposed in [DWE16]. A global occupancy map is kept. A new logistic regression classifier is trained for every new observation, and the

local maps can be fused by overlapping Hilbert maps.

2.3.3 Prediction

The objective of the prediction is to estimate unobserved space from the observations of observed space. GPOMs can be applied to predict the occupancy probability of unobserved space. In [CLLH07], the prediction-based SLAM algorithm (P-SLAM) is proposed to predict the structure of unobserved space. The structures of the unobserved space can be predicted by matching the structures surrounding the target space and the collected structures of observed space. This approach works well in the environments that have repeated structures. In [SNS15], unexplored areas are predicted by finding similarities between the current surroundings of the target space and the previously built maps. This method can help robots to obtain more information about unobserved space and speed up the loop closure.

2.4 Mapping in dynamic environments

2.4.1 Object tracking

The main difference between static and dynamic environments is the existence of dynamic obstacles. Dynamic objects can be roughly divided into two categories: low and high dynamic objects. The robot object mapping algorithm (ROMA) proposed in [BLST02] considers low dynamic objects. Many static occupancy grid maps are learnt with regular time intervals and the low dynamic objects are detected by differencing these static occupancy grid maps. The EM algorithm is applied to learn models of dynamic objects. The environment is described by two different occupancy grid maps in [WS04]. One map models the static environment, and the other one models dynamic objects. The dynamic map indicates objects with high or low dynamics. In [MT07], a temporal occupancy grid is proposed for dynamic environments. The whole history with different time indexes is stored. Static and dynamic objects can be extracted from these temporal occupancy grid maps. Cells with low dynamics and high probabilities of being occupied are regarded as static. Dynamic and static objects is classified based on phase congruency in [HDLF19]. Low dynamic objects, such as the doors, do not change too much. Their possible configurations can be extracted. For high dynamic objects, a high dynamic map is generated. For parking environments, a method to detect both parallel-parked and cross-parked vehicles is proposed in [DHS+14]. An occupancy grid map is regarded as an image, and vehicle candidates are extracted from the occupancy map. Two random forest classifiers are trained to classify these vehicle candidates.

The detection and tracking of moving objects (DATMO) approach tries to estimate the number of potential targets, their positions, and velocities from sensor data. The SLAM problem and the DATMO problem are considered simultaneously in [WT02] and [WTT+07]. In [BH10], a robust algorithm to detect dynamic obstacles from occupancy maps is presented. Grid cells are associated with the state of an object. In [TTWB14], a uniform, grid-based representation is proposed. For every grid cell, its velocity is estimated by a particle filter. Static and dynamic grid cells are classified based on their velocity distributions. Extended objects tracking approach based on object-local occupancy grid maps is proposed in [SADD14] where a particle filter is used to estimate both the dynamic states of objects and their static shapes. The particles of each tracked object contain a part describing its dynamic state and a part representing its shape. In [WU18], the cell dynamics are solved by standard tracking filters and velocities are estimated indirectly.

Bayesian occupancy filtering (BOF) [CPL+06] is a Bayesian framework for dynamic environments. Environments are also represented by occupancy grid maps. Each grid cell is described by a two-dimensional spatial position and a two-dimensional velocity. With velocity information, the BOF can predict the spread

of spatial occupancy and avoid the collision with partially observed moving obstacles. This work is improved to reduce its complexities in [CTLM06]. The BOF is applied to fuse measurements from different sensors and extract the objects. Based on the velocity estimation in the BOF, the position and velocity of each occupancy grid cell can be updated by a Kalman filter. Similarly, the BOF is used to fuse the information provided by IR sensors and CCD cameras [RM09]. A clustering-tracking algorithm is applied to extract and track objects based on the BOF output in [MMRL08]. A sequential Monte Carlo (SMC) implementation of the BOF is proposed in [DON11]. The distribution of each grid cell is approximated by a number of particles with weights obtained by fusing measurement data. This work is implemented to fuse laser and radar sensor data in [NYK+15]. Hybrid Sampling BOF is proposed in [NRL14]. Occupancy grid maps are used to represent the static part, while the dynamic part is modelled by a set of moving particles.

2.4.2 Markov models in dynamic mapping

In dynamic environments, the state of one point may change over time. Under the assumption that its current state depends on the previous state, Markov models can be applied to represent the dynamic behaviour. The hidden Markov model (HMM) [Rab89] is applied in [MDBB12] and grid maps are used to represent dynamic environments. The state change of one grid cell is represented by a state transition matrix, which can be used to predict the possible state in the future. The state transition matrix is estimated by an online learning algorithm [MD08]. Normally, the well-known Baum–Welch algorithm is applied to learn the transition matrix of an HMM, which consists of a forward procedure and a backward procedure. In online methods, only the forward procedure is considered. In an HMM, there are two possible states: occupied and free. The grid cells in the area covered by measurement ranges have two possible observations: occupied and free. The grid cells outside measurement ranges are labelled as no-observation. In [RDH+16], there are three possible observations: true, false and not observable. The underlying stochastic Markov process consists of seven components: “true”, “false”, “unknown”, “dynamic”, “falsely false”, “falsely true”, “falsely true/false”. The last three are used to deal with wrong observations.

The input-output HMM (IOHMM) [BF95] is an extension to the HMM. In an IOHMM, observations and transition probabilities are conditional on the input sequence. It is applied to model the motion patterns of dynamic objects in [WAJF14]. Every grid cell is assigned an IOHMM. The input of an IOHMM is the observations of neighbouring cells in the previous time step. In this manner, the spatial correlation is taken into account.

The Explicit-state-duration HMM (EDHMM) [DWW12] (or, equivalently, the hidden semi-Markov model (HSMM) [Yu10]) is another extension of the HMM. In an EDHMM, the underlying process is a semi-Markov chain with variable duration. EDHMM is applied to differentiate dynamic cells from static environment in [DKS15]. For every grid cell, there are two latent states: dynamic and static. The observable states considered are also occupied and free.

2.5 Summary

In summary, range finders directly provide their distances to surrounding objects, and different kinds of maps can be built for static and dynamic environments. Occupancy maps are classical representations for static environments. Because of the uncertainty in sensor models or the error in match methods, the built maps may be noisy. MRFs and Gaussian processes (or Gaussian random fields) can be applied to deal with the inconsistency. However, these methods have high computational complexity. For dynamic environments, one popular technique is to estimate the positions and velocities of dynamic objects. The other one is to use Markov models to model the dynamic objects and the other space. Normally the

inconsistency in dynamic environments is not considered. This thesis uses a different representation for static environments to reduce computational complexity. The two kinds of random fields are also applied to built smooth maps for dynamic environments.

3

Preliminaries

This thesis, as most state-of-the-art algorithms for robotic mapping, is developed based on probability theory. The probabilistic materials relevant to the work in this thesis are presented.

3.1 Probabilistic estimation

3.1.1 Bayes estimation

Given two random variables X and Y, and the corresponding probability density functions, the Bayes rule [Bil08] is given by

$$p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)}, \tag{3.1}$$

where $p(y) > 0$. The probability $p(y \mid x)$ is called likelihood function of x. The distribution $p(x)$ is called prior and the distribution $p(x \mid y)$ after correction using the observations is called the posterior distribution. The probability $p(y)$ is not a function of x and has the role of a normalizer.

3.1.2 Maximum a posterior

The Bayes rule computes the complete posterior probability distribution of x. If an estimation of x requires a crisp value to be selected, a criteria has to be employed to find such a value. The maximum a posterior (MAP) as equation (3.3) is one such criteria. It just selects the x for which the posterior distribution $p(x \mid y)$ attains its maximum value.

$$\hat{x} = \arg\max_x p(x \mid y) \tag{3.2}$$

$$= \arg\max_x p(y \mid x)p(x). \tag{3.3}$$

Since $p(y)$ is just a normalizing constant, it does not affect the result of the MAP estimation and can be omitted.

3.1.3 Maximum likelihood

Another estimation method called maximizing likelihood (ML), favoured in frequentist approach to probabilities, does not involve a prior. Here, only the likelihood function is used as

$$\hat{x} = \arg\max_x p(y \mid x). \tag{3.4}$$

In some cases, when a uniform distribution can be used as a prior, the MAP and ML methods give the same results.

3.2 Stochastic processes

A stochastic process is a collection of random variables $X(t)$ usually indexed by time $t \in T$, where T is a collection of discrete time instants, or a time interval. In a stochastic process, at any point t, $X(t)$ is a random variable. There are many kinds of stochastic processes. Gaussian processes and Markov models are introduced next.

3.2.1 Gaussian processes and Gaussian random fields

A Gaussian process [RW06] is a particular sample of a stochastic process. It is used to describe a probability distribution over functions in a continuous domain, such as time or space. Assuming that $X(t)$ is a Gaussian process in time domain, at every time t, $X(t)$ is a Gaussian distributed random variable. The Gaussian process allows multiple points in time to be considered. In that case, selecting ζ points $X(t_1), \cdots, X(t_\zeta)$, the random variables will be multivariate Gaussian distributed with mean vector μ and covariance matrix K. The Gaussian process $X(t)$ is denoted by

$$X(t) \sim \mathcal{GP}(\mu(t), \mathcal{C}(t, t')), \tag{3.5}$$

where $\mu(t)$ is a mean function

$$\mu(t) = E[X(t)], \tag{3.6}$$

and $\mathcal{C}(t, t')$ is a covariance function

$$\mathcal{C}(t, t') = E[(X(t) - \mu(t))(X(t') - \mu(t'))], \tag{3.7}$$

which is also called kernel that computes the covariance components that make up the covariance matrix K. Denoting a time sequence by $T = (t_1, \cdots, t_\zeta)$, the corresponding random variable vector $\boldsymbol{X} = [X(t_1), \cdots, X(t_\zeta)]^\top$ is Gaussian distributed as

$$\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{K}), \tag{3.8}$$

where the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{K}$ are given by

$$\boldsymbol{\mu} = \begin{bmatrix} \mu(t_1) & \dots & \mu(t_\zeta) \end{bmatrix}^\top,$$

$$\boldsymbol{K} = \begin{bmatrix} \mathcal{C}(t_1, t_1) & \dots & \mathcal{C}(t_1, t_\zeta) \\ \vdots & & \vdots \\ \mathcal{C}(t_1, t_\zeta) & \dots & \mathcal{C}(t_\zeta, t_\zeta) \end{bmatrix}.$$

There are several common covariance functions, such as

- Constant: $\mathcal{C}(t, t') = c$,
- Gaussian noise: $\mathcal{C}(t, t') = \sigma^2 \mathbf{1}_{tt'}$,
- Squared exponential: $\mathcal{C}(t, t') = \exp\left(-\frac{|t-t'|^2}{2\ell^2}\right)$,
- Ornstein–Uhlenbeck: $\mathcal{C}(t, t') = \exp\left(-\frac{|t-t'|}{\ell}\right)$,
- Triangular: $\mathcal{C}(t, t') = 1 - \frac{|t-t'|}{\ell}$ for $|t - t'| \leq \ell$ and 0 otherwise,

where $\mathbf{1}_{tt'}$ is the indicator function (1 when $t = t'$, 0 otherwise) and ℓ is the characteristic length-scale of the process. In the first covariance function, the covariance between any two points is a constant c. In the second covariance function, different points are uncorrelated. In other words, one point is only correlated with itself. This function is useful to model additive Gaussian noise. In the last three covariance functions, the covariance between two points becomes smaller as the distance increases. For large distances, the covariance is 0 or close to 0.

Gaussian processes can be used to predict unknown points given some training data. The idea is to derive the distribution of unknown points conditioning on training data from their joint distribution. Assuming the outputs of $\boldsymbol{X}$ are observed without noise and selecting some unknown points denoted by a random variable vector $\boldsymbol{X}_*$ that is Gaussian distributed with mean vector $\boldsymbol{\mu}_*$ and covariance matrix $\boldsymbol{K}_{**}$, the joint distribution of training points $\boldsymbol{X}$ and test points $\boldsymbol{X}_*$ is Gaussian distributed as

$$\begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{X}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} & \boldsymbol{K}_{T*}^\top \\ \boldsymbol{K}_{T*} & \boldsymbol{K}_{**} \end{bmatrix}\right), \tag{3.9}$$

where $\boldsymbol{K}_{T*}$ is the covariance matrix between $\boldsymbol{X}$ and $\boldsymbol{X}_*$. The conditional distribution of $\boldsymbol{X}_*$ [RW06] is

$$\boldsymbol{X}_* | \boldsymbol{X} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_*, \hat{\boldsymbol{K}}_*), \tag{3.10}$$

where the predictive mean vector $\hat{\boldsymbol{\mu}}_*$ and the predictive covariance matrix $\hat{\boldsymbol{K}}_*$ are given by

$$\hat{\boldsymbol{\mu}}_* = \boldsymbol{\mu}_* + \boldsymbol{K}_{T*}\boldsymbol{K}^{-1}(\boldsymbol{X} - \boldsymbol{\mu}), \tag{3.11}$$

$$\hat{\boldsymbol{K}}_* = \boldsymbol{K}_{**} - \boldsymbol{K}_{T*}\boldsymbol{K}^{-1}\boldsymbol{K}_{T*}^{\top}. \tag{3.12}$$

Normally the outputs of a Gaussian process are observed with noise. Assuming the noisy outputs of $X(t)$ are

$$Y(t) = X(t) + \epsilon, \tag{3.13}$$

where ϵ is assumed to be additive Gaussian noise with zero mean and variance σ^2, the noisy output vector of $\boldsymbol{X}$ denoted by $\boldsymbol{Y}$ is Gaussian distributed with mean vector μ and covariance matrix $\boldsymbol{K} + \sigma^2\boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. The joint distribution of noisy training points $\boldsymbol{Y}$ and test points $\boldsymbol{X}_*$ is

$$\begin{bmatrix} \boldsymbol{Y} \\ \boldsymbol{X}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} + \sigma^2\boldsymbol{I} & \boldsymbol{K}_{T*}^{\top} \\ \boldsymbol{K}_{T*} & \boldsymbol{K}_{**} \end{bmatrix} \right). \tag{3.14}$$

The predictive conditional distribution becomes

$$\boldsymbol{X}_* | \boldsymbol{Y} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_*, \hat{\boldsymbol{K}}_*), \tag{3.15}$$

where the predictive mean vector $\hat{\boldsymbol{\mu}}_*$ and the predictive covariance matrix $\hat{\boldsymbol{K}}_*$ are rewritten as

$$\hat{\boldsymbol{\mu}}_* = \boldsymbol{\mu}_* + \boldsymbol{K}_{T*}(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}(\boldsymbol{Y} - \boldsymbol{\mu}), \tag{3.16}$$

$$\hat{\boldsymbol{K}}_* = \boldsymbol{K}_{**} - \boldsymbol{K}_{T*}(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{K}_{T*}^{\top}. \tag{3.17}$$

The main disadvantage in this prediction is the need to compute the inverse of the covariance matrix. If there are more training points, the inverse matrix should be recomputed.

A Gaussian random field (GRF) is a Gaussian process where random variables are indexed by a multi-dimensional domain, such as space. The inputs of the corresponding covariance functions are distances of two random variables in the multi-dimensional domain. Except for the covariance functions, the other equations in this section can be used in GRFs.

3.2.2 Markov chains

A Markov chain is a stochastic process that satisfies the so called Markov property. The Markov property states the future is conditionally independent of the past given the present,

$$p(x_{t+1} \mid x_t, \cdots, x_0) = p(x_{t+1} \mid x_t). \tag{3.18}$$

Here it is assumed that the random variables are in the discrete-time domain, are denoted by X_t with time index t, and have the same finite state space $\{s_1, s_2, \cdots, s_g\}$ with g states. A Markov chain is shown in Figure 3.1. The state transition probabilities are represented in a transition matrix

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & \cdots & a_{1g} \\ \vdots & \ddots & \vdots \\ a_{g1} & \cdots & a_{gg} \end{bmatrix},$$

where a_{ij} is the probability of transitioning from state s_i to s_j. The transition matrix $\boldsymbol{A}$ is assumed to be time-invariant.

Multiplying the current state distribution with $\boldsymbol{A}$ gives the state distribution at the next time step. When

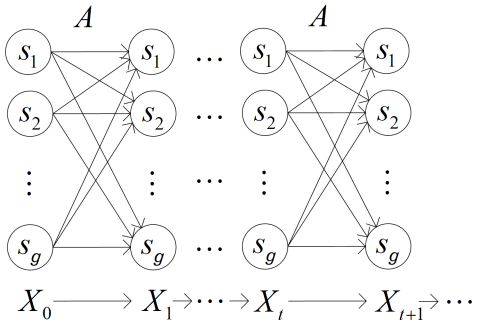


Figure 3.1: An example of a Markov chain. There are g possible states and the transition matrix is denoted by $\boldsymbol{A}$.

these two distributions are the same, it means the state distribution in the future does not change anymore. This distribution is called stationary distribution. Given the transition matrix $\boldsymbol{A}$, the stationary distribution denoted by a vector $\boldsymbol{\mathcal{G}} = [\mathcal{G}_1, \cdots, \mathcal{G}_g]$ can be found by solving

$$\boldsymbol{\mathcal{G}}\boldsymbol{A} = \boldsymbol{\mathcal{G}} \tag{3.19}$$

with the additional constraint that elements must sum to 1.

The probability of changing state s_i to the others is $1 - a_{ii}$, where a_{ii} is the probability of staying in state s_i. The probability of staying in state s_i for d time steps from time step t is

$$p(s_i, d) = (1 - a_{ii}) a_{ii}^{d-1} p(x_t = s_i), \tag{3.20}$$

where $p(x_t = s_i)$ is the probability of being in state s_i at time t. The overall expected duration [Rat02] is defined by

$$\begin{aligned} E(d) &= \sum_{s_i} \sum_{d} d p(s_i, d) \\ &= \sum_{i} \frac{1}{1 - a_{ii}} p(x_t = s_i), \end{aligned} \tag{3.21}$$

which can be used to indicate how dynamic a process is. Low dynamic processes have long overall expected durations.

3.2.3 Hidden Markov models

A hidden Markov model (HMM) [Rab89] is a Markov chain for which the states cannot be observed directly but only though an observation process which is itself stochastic. Here only the HMM in the discrete-time domain is considered, which is assume to have a finite observation space $\{o_1, o_2, \cdots, o_q\}$ with q possible observations. A graphical model of an HMM is shown in Figure 3.2. The observation model is represented by emission probabilities $p(y_t \mid x_t)$, which make up an emission matrix

$$\boldsymbol{B} = \begin{bmatrix} b_{11} & \dots & b_{1q} \\ \vdots & \ddots & \vdots \\ b_{g1} & \dots & b_{gq} \end{bmatrix},$$

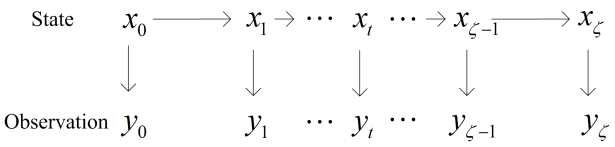


Figure 3.2: Graphical model of an HMM

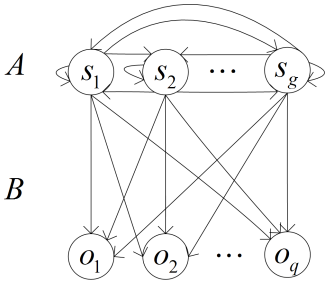


Figure 3.3: Probabilistic parameters of an HMM. There are q possible observations and the emission matrix is denoted by $\boldsymbol{B}$.

where b_{ij} is the probability of observing o_j given state s_i. The probabilistic parameters of an HMM is shown in Figure 3.3.

If the parameters of an HMM are unknown, they have to be estimated from the data. Given an observation sequence, the parameters of an HMM can be estimated. Meanwhile the initial distribution $\rho_i = p(x_0 = s_i)$ of the Markov chain should be estimated together with the HMM parameters. An observation sequence is denoted by $O = (y_0, y_1, \cdots, y_\zeta)$ and all the parameters are denoted by $\theta = (\rho, \boldsymbol{A}, \boldsymbol{B})$, where $\rho = \{\rho_i\}$ is the initial state distribution. If there is no prior knowledge on θ, the problem can be solved by maximum likelihood. The likelihood function is

$$p(O \mid \theta) = \sum_{\mathcal{X}} p(O, \mathcal{X} \mid \theta), \tag{3.22}$$

where $\mathcal{X} = (x_0, x_1, \cdots, x_\zeta)$ is an underlying state sequence and

$$p(O, \mathcal{X} \mid \theta) = \rho_i b_{x_0 y_0} \prod_{t=1}^{\zeta} a_{x_{t-1} x_t} b_{x_t y_t}. \tag{3.23}$$

It is infeasible to maximize the likelihood function directly. A common alternative is to use the Baum–Welch algorithm [Tu15] to recursively maximize a function

$$\begin{aligned} Q(\theta, \theta^{(k)}) &= \sum_{\mathcal{X}} p(\mathcal{X} \mid O, \theta^{(k)}) \log p(\mathcal{X}, O \mid \theta) && (3.24) \\ &= \sum_{\mathcal{X}} p(\mathcal{X} \mid O, \theta^{(k)}) \left(\log \rho_{x_0} + \sum_{t=1}^{\zeta} \log a_{x_{t-1} x_t} + \sum_{t=0}^{\zeta} \log b_{x_t y_t} \right) \\ &= \sum_i \gamma_i(0) \log \rho_{s_i} + \sum_{t=1}^{\zeta} \sum_{i,j} \xi_{ij}(t) \log a_{ij} + \sum_{t=0}^{\zeta} \sum_i \gamma_i(t) \log b_{i y_t}, && (3.25) \end{aligned}$$

where the variable $\gamma_i(t) = p(x_t = s_i \mid O, \theta^{(k)})$ is the probability of being in state s_i at time t given the observation sequence O and the parameters $\theta^{(k)}$, the variable $\xi_{ij}(t) = p(x_t = s_i, x_{t+1} = s_j \mid O, \theta^{(k)})$

is the probability of being in state s_i at time t and s_j at time $t+1$ given the observation sequence O and parameters $\theta^{(k)}$. Uniting like terms in the second line gives equation (3.25). This algorithm can be described as:

1. Compute $Q(\theta, \theta^{(k)})$,
2. Set $\theta^{(k+1)} = \arg\max_\theta Q(\theta, \theta^{(k)})$.

The two steps should be repeated until convergence. Maximizing $Q(\theta, \theta^{(k)})$ gives the estimations of the parameters,

$$\rho_i^{(k+1)} = \gamma_i(0), \tag{3.26}$$

$$a_{ij}^{(k+1)} = \frac{\sum_{t=1}^{\zeta} \xi_{ij}(t)}{\sum_{t=1}^{\zeta} \gamma_i(t)}, \tag{3.27}$$

$$b_{ij}^{(k+1)} = \frac{\sum_{t=0}^{\zeta} \delta(y_t, o_j)\gamma_i(t)}{\sum_{t=0}^{\zeta} \gamma_i(t)}, \tag{3.28}$$

where $\delta(y_t, o_j)$ is the Kronecker delta. Before computing $\gamma_i(t)$ and $\xi_{ij}(t)$, another two probabilities are required. The first one is $\alpha_i(t) = p(y_0, y_1, \cdots, y_t, x_t = s_i \mid \theta)$, which is the probability of seeing the $y_0, y_1, \cdots, y_t$ and being in state s_i at time t. This step called forward procedure is computed recursively from time 0 to t,

$$\alpha_i(0) = \rho_i b_{iy_0}, \tag{3.29}$$

$$\alpha_j(t+1) = b_{jy_{t+1}} \sum_{i=1}^{g} \alpha_i(t) a_{ij}. \tag{3.30}$$

The second one is $\beta_i(t) = p(y_{t+1}, \cdots, y_\zeta \mid x_t = s_i, \theta)$, which is the probability of the ending partial sequence $y_{t+1}, \cdots, y_\zeta$ given starting state s_i at time t. This step called backward procedure is calculated from time ζ to t,

$$\beta_i(\zeta) = 1, \tag{3.31}$$

$$\beta_i(t) = \sum_{j=1}^{g} \beta_j(t+1) a_{ij} b_{jy_{t+1}}. \tag{3.32}$$

According to the Bayes rule, the variables $\gamma_i(t)$ and $\xi_{ij}(t)$ are given by

$$\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{g} \alpha_j(t)\beta_j(t)}, \tag{3.33}$$

$$\xi_{ij}(t) = \frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_{jy_{t+1}}}{\sum_{i=1}^{g} \sum_{j=1}^{g} \alpha_i(t) a_{ij} \beta_j(t+1) b_{jy_{t+1}}}. \tag{3.34}$$

3.2.4 Markov random fields

A Markov random field (MRF) [Li09] is a multi-dimensional stochastic process where the random variables satisfy another Markov property. This Markov property states a random variable X_i is conditionally independent of all other variables given its neighbours whose Euclidean distances to X_i are not more than a radius r. Assuming the index set of random variables is $\mathcal{S} = \{1, 2, \ldots, n\}$, the index set of the neighbours of X_i is

$$N_i = \{i' \in \mathcal{S} \setminus i \mid \mathrm{dis}(X_i, X_{i'}) \leq r\}, \tag{3.35}$$

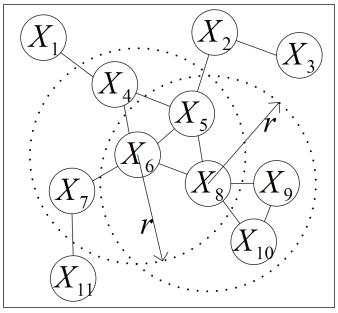


Figure 3.4: An example of neighbours. Two variables connected by a line are neighbours.

	$\ddots$	$\vdots$	$\iddots$	
$\cdots$	X_{i-1}	X_i	X_{i+1}	$\cdots$
	$\iddots$	$\vdots$	$\ddots$	

(a) A regular MRF

5	4	3	4	5
4	2	1	2	4
3	1	X_i	1	3
4	2	1	2	4
5	4	3	4	5

(b) Regular neighbourhood

Figure 3.5: Neighbourhood in a regular MRF [Li09]

where $\mathcal{S} \setminus i$ means the set $\mathcal{S}$ without i. An example of neighbours is shown in Figure 3.4 where all the neighbours are connected by lines. For different random variables, the numbers of neighbours may be different. The Markov property is given by

$$p(x_i \mid x_{\mathcal{S} \setminus i}) = p(x_i \mid x_{N_i}). \tag{3.36}$$

A clique c is defined as a subset of random variables, in which every two random variables are neighbours. A random variable X_i is a single clique. A pair-variable clique consists of two random variables connected by a line, as shown in Figure 3.4. If there are more than two random variables in a clique, these random variables are neighbours to one another, such as cliques $\{X_4, X_5, X_6\}$, $\{X_5, X_6, X_8\}$, and $\{X_8, X_9, X_{10}\}$.

As shown in Figure 3.5(a), neighbours are symmetrical in a regular MRF. A first-order neighbourhood consists of the nearest neighbours labelled as 1 in Figure 3.5(b). The other labels are assigned to the outermost neighbours in the corresponding order neighbourhood. The high order neighbourhood contains all the neighbours in the lower order neighbourhoods, such as a second-order neighbourhood consisting of neighbours labelled as 1 and 2. As shown in Figure 3.6, the cliques in a second-order neighbourhood consist of single cliques, pair-variable cliques, triple-variable cliques and quadruple-variable cliques.

Assuming the collection of all the cliques is denoted by C and the vector of all the random variables in an MRF is denoted by X, the joint distribution of $\boldsymbol{X}$ is

$$p(\boldsymbol{x}) = \frac{1}{Z} \exp\left(-\frac{U(\boldsymbol{x})}{\mathcal{T}}\right), \tag{3.37}$$

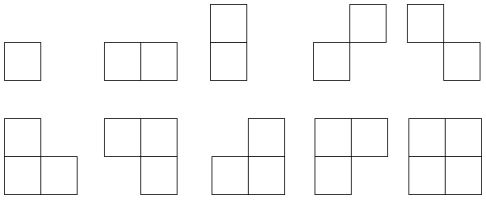


Figure 3.6: Cliques in a second-order neighbourhood [Li09]

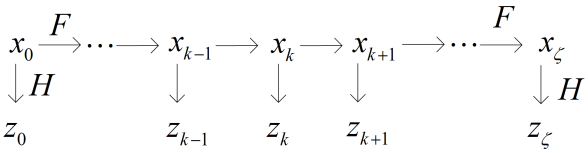


Figure 3.7: An example of UKF

where the normalizing constant Z called partition function is given by

$$Z = \sum_{\boldsymbol{x}} \exp\left(-\frac{U(\boldsymbol{x})}{\mathcal{T}}\right). \tag{3.38}$$

The energy function $U(\boldsymbol{x})$ is a sum of all the clique potentials V_c,

$$U(\boldsymbol{x}) = \sum_{c \in C} V_c, \tag{3.39}$$

where V_c is a function of the values of random variables in the corresponding clique. The constant $\mathcal{T}$, called the temperature, is a constant of the distribution.

3.3 Unscented Kalman filters

3.3.1 Filter

Unscented Kalman filters (UKFs) [WVDM00] are a class of methods used to estimate the states of a nonlinear dynamic system. Different from HMMs, the states take values in a continuous space. A graphical model underlying the UKF is shown in Figure 3.7. The nonlinear system is specified by a state-space model discretized in the time domain,

$$\boldsymbol{x}_k = F(\boldsymbol{x}_{k-1}) + \mathcal{W}, \tag{3.40}$$

where $\boldsymbol{x}_k$ is the state vector and $\mathcal{W}$ is process noise assumed to be Gaussian distributed with zero mean and covariance $\boldsymbol{\mathcal{R}}$. Given a current state, the future states can be predicted. The nonlinear measurement model is given by

$$\boldsymbol{z}_k = H(\boldsymbol{x}_k) + \mathcal{V}, \tag{3.41}$$

where $\mathcal{V}$ is measurement noise assumed to be Gaussian distributed with zero mean and covariance $\boldsymbol{\mathcal{Q}}$.

In probabilistic form, the objective of the UKF is to obtain the distribution $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:k})$ which characterizes the current state distribution conditioned on all the measurements already obtained. There are two steps:

state prediction and correction. In state prediction step, the objective is to estimate $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:k-1})$,

$$p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:k-1}) = \int p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}, \boldsymbol{z}_{0:k-1}) p(\boldsymbol{x}_{k-1} \mid \boldsymbol{z}_{0:k-1}) \mathrm{d}\boldsymbol{x}_{k-1} \tag{3.42}$$

$$= \int p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}) p(\boldsymbol{x}_{k-1} \mid \boldsymbol{z}_{0:k-1}) \mathrm{d}\boldsymbol{x}_{k-1}, \tag{3.43}$$

where $p(\boldsymbol{x}_{k-1} \mid \boldsymbol{z}_{0:k-1})$ is the estimation of the previous state $\boldsymbol{x}_{k-1}$ given all the past measurements $\boldsymbol{z}_{0:k-1}$. The current state $\boldsymbol{x}_k$ only depends on the previous state $\boldsymbol{x}_{k-1}$. If the system is linear with Gaussian noise, it is easy to predict the next state. For the nonlinear system, this step can be done by the scaled unscented transformation [Jul02] where some sigma points are chosen from the distribution $p(\boldsymbol{x}_{k-1} \mid \boldsymbol{z}_{0:k-1})$ and projected through the system model $p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})$. The number of sigma points chosen is $2L+1$, where L is the dimension of the state vector $\boldsymbol{x}_k$. Assuming the distribution $p(\boldsymbol{x}_{k-1} \mid \boldsymbol{z}_{0:k-1})$ is Gaussian distributed with mean vector $\bar{\boldsymbol{x}}_{k-1}^{+}$ and covariance matrix $\boldsymbol{P}_{k-1}^{+}$, the chosen sigma points $\boldsymbol{x}_{k-1}^{i+}$ are

$$\begin{aligned}
\boldsymbol{x}_{k-1}^{0+} &= \bar{\boldsymbol{x}}_{k-1}^{+}, \\
\boldsymbol{x}_{k-1}^{i+} &= \bar{\boldsymbol{x}}_{k-1}^{+} + (\sqrt{(L+\lambda)\boldsymbol{P}_{k-1}^{+}})_i \ \text{ for } \ i = 1, \cdots, L, \\
\boldsymbol{x}_{k-1}^{i+} &= \bar{\boldsymbol{x}}_{k-1}^{+} - (\sqrt{(L+\lambda)\boldsymbol{P}_{k-1}^{+}})_{i-L} \ \text{for } \ i = (L+1), \cdots, 2L.
\end{aligned} \tag{3.44}$$

The corresponding mean weights w_m^i and covariance weights w_c^i are given by

$$\begin{aligned}
w_m^0 &= \frac{\lambda}{L+\lambda}, \\
w_c^0 &= \frac{\lambda}{L+\lambda} + 1 - \alpha^2 + \beta, \\
w_m^i &= w_c^i = \frac{1}{2(L+\lambda)} \quad i = 1, \cdots, 2L,
\end{aligned} \tag{3.45}$$

where α, β, and κ are parameters, the variable $\lambda = \alpha^2(L+\kappa) - L$. The formula $(\sqrt{(L+\lambda)\boldsymbol{P}_{k-1}^{+}})_i$ is the i_{th} column of $\sqrt{(L+\lambda)\boldsymbol{P}_{k-1}^{+}}$. The parameter α is a positive scaling parameter and its range is (0,1). The parameter κ is a non-negative scaling parameter and its range is [0, ∞). Normally κ is set to 0 [VDM04]. The parameter β is used to incorporate prior knowledge of the distribution of x_{k-1}. For Gaussian distributions, the optimal choice is 2 [Jul02]. The mean and variance of the distribution $p(\boldsymbol{x}_{k-1} \mid \boldsymbol{z}_{0:k-1})$ are given by

$$\bar{\boldsymbol{x}}_{k-1}^{+} = \sum_{i=0}^{2L} w_m^i \boldsymbol{x}_{k-1}^{i+}, \tag{3.46}$$

$$\boldsymbol{P}_{k-1}^{+} = \sum_{i=0}^{2L} w_c^i (\boldsymbol{x}_{k-1}^{i+} - \bar{\boldsymbol{x}}_{k-1}^{+})(x_{k-1}^{i+} - \bar{\boldsymbol{x}}_{k-1}^{+})^{\top}. \tag{3.47}$$

For every sigma point $\boldsymbol{x}_{k-1}^{i+}$, a corresponding point $\boldsymbol{x}_k^i$ can be obtained by transforming $\boldsymbol{x}_{k-1}^{i+}$ through the system model. The mean vector and covariance matrix of the distribution $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:k-1})$ can be respectively approximated by

$$\bar{\boldsymbol{x}}_k^{-} \approx \sum_{i=0}^{2L} w_m^i \boldsymbol{x}_k^i, \tag{3.48}$$

$$\boldsymbol{P}_k^- \approx \sum_{i=0}^{2L} w_c^i (\boldsymbol{x}_k^i - \bar{\boldsymbol{x}}_k^-)(\boldsymbol{x}_k^i - \bar{\boldsymbol{x}}_k^-)^\top + \boldsymbol{\mathcal{R}}. \tag{3.49}$$

The objective of the correction step is to estimate $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:k})$ given the current measurement $\boldsymbol{z}_k$,

$$p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:k}) = \frac{p(\boldsymbol{z}_k \mid \boldsymbol{x}_k) p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:k-1})}{p(\boldsymbol{z}_k \mid \boldsymbol{z}_{0:k-1})}. \tag{3.50}$$

First, the predicted measurement is obtained by projecting the sigma points chosen from distribution $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:k-1})$ through the measurement model. Since the sigma points $\boldsymbol{x}_k^i$ with weights w_m^i and w_c^i do not incorporate the system uncertainty $\mathcal{W}$, they should be augmented [VDM04]. Assuming the new sigma points are denoted by $\boldsymbol{x}_k^{i-}$, the first sigma point $\boldsymbol{x}_k^{0-} = \boldsymbol{x}_k^0$ does not change, and the other sigma points are augmented as

$$\boldsymbol{x}_k^{i-} = \begin{cases} \boldsymbol{x}_k^i + (\sqrt{(L+\lambda)\boldsymbol{\mathcal{R}}})_i & \text{for } i = 1, \cdots, L, \\ \boldsymbol{x}_k^i - (\sqrt{(L+\lambda)\boldsymbol{\mathcal{R}}})_{i-L} & \text{for } i = (L+1), \cdots, 2L. \end{cases} \tag{3.51}$$

The mean weight and covariance weight for each sigma point do not change. Assuming the new sigma points after projecting $\boldsymbol{x}_k^{i-}$ through the measurement model are denoted by $\boldsymbol{z}_k^i$, the mean vector and covariance matrix of the predicted measurement can be respectively approximated by

$$\bar{\boldsymbol{z}}_k \approx \sum_{i=0}^{2L} w_m^i \boldsymbol{z}_k^i, \tag{3.52}$$

and

$$\boldsymbol{P}_k^z \approx \sum_{i=0}^{2L} w_c^i (\boldsymbol{z}_k^i - \bar{\boldsymbol{z}}_k)(\boldsymbol{z}_k^i - \bar{\boldsymbol{z}}_k)^\top + \boldsymbol{\mathcal{Q}}. \tag{3.53}$$

Based on the predicted measurement, the mean vector and covariance matrix of the new estimation $\boldsymbol{x}_k^+$ are finally calculated by

$$\bar{\boldsymbol{x}}_k^+ = \bar{\boldsymbol{x}}_k^- + \boldsymbol{\mathcal{K}}(\boldsymbol{z}_k - \bar{\boldsymbol{z}}_k), \tag{3.54}$$

$$\boldsymbol{P}_k^+ = \bar{\boldsymbol{P}}_k^- - \boldsymbol{\mathcal{K}} \boldsymbol{P}_k^z \boldsymbol{\mathcal{K}}^\top, \tag{3.55}$$

where the Kalman gain $\boldsymbol{\mathcal{K}}$ and the cross covariance $\boldsymbol{P}_{xz}$ are given by

$$\boldsymbol{\mathcal{K}} = \boldsymbol{P}_{xz} \boldsymbol{P}_k^z, \tag{3.56}$$

$$\boldsymbol{P}_{xz} = \sum_{i=0}^{2L} w_c^i (\boldsymbol{x}_k^i - \bar{\boldsymbol{x}}_k)(\boldsymbol{z}_k^i - \bar{\boldsymbol{z}}_k)^\top. \tag{3.57}$$

3.3.2 Smoothing

Smoothing is a little different from the filter described above. For the same graphical model shown in Figure 3.7, the objective of the smoothing problem is to estimate $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:\zeta})$ where future measurements are also used to estimate the current state $\boldsymbol{x}_k$. This problem can be solved by

$$p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:\zeta}) = \frac{p(\boldsymbol{z}_{k+1:\zeta} \mid \boldsymbol{x}_k) p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:k})}{p(\boldsymbol{z}_{k+1:\zeta} \mid \boldsymbol{z}_{0:k})}. \tag{3.58}$$

The solution can also be divided into two steps: a forward step $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:k})$ and a backward step $p(\boldsymbol{z}_{k+1:\zeta} \mid \boldsymbol{x}_k)$.

One simple case of smoothing is the estimation $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:k+1})$ of the current state $\boldsymbol{x}_k$ using a single future measurement $\boldsymbol{z}_{k+1}$ and given by

$$p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:k+1}) = \frac{p(\boldsymbol{z}_{k+1} \mid \boldsymbol{x}_k)p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:k})}{p(\boldsymbol{z}_{k+1} \mid \boldsymbol{z}_{0:k})}. \tag{3.59}$$

The forward step estimates $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:k})$, which is obtained in the previous section. The backward step computes $p(\boldsymbol{z}_{k+1} \mid \boldsymbol{x}_k)$ which is given by

$$p(\boldsymbol{z}_{k+1} \mid \boldsymbol{x}_k) = \int p(\boldsymbol{z}_{k+1} \mid \boldsymbol{x}_{k+1})p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)\mathrm{d}\boldsymbol{x}_k. \tag{3.60}$$

The sigma points corresponding to the prediction of the measurement $\boldsymbol{z}_{k+1}$ can be obtained by the same method based on the scaled unscented transformation and are denoted by $\boldsymbol{z}^i_{k+1}$. The mean vector and covariance matrix of this predicted measurement are respectively approximated by

$$\bar{\boldsymbol{z}}_{k+1} \approx \sum_{i=0}^{2L} w^i_m \boldsymbol{z}^i_{k+1}, \tag{3.61}$$

and

$$\boldsymbol{P}^z_{k+1} \approx \sum_{i=0}^{2L} w^i_c (\boldsymbol{z}^i_{k+1} - \bar{\boldsymbol{z}}_{k+1})(\boldsymbol{z}^i_{k+1} - \bar{\boldsymbol{z}}_{k+1})^\top + \boldsymbol{\mathcal{Q}}. \tag{3.62}$$

The mean vector and covariance matrix of the distribution $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:k+1})$ are respectively given by

$$\bar{\boldsymbol{x}}^{++}_k = \bar{\boldsymbol{x}}^{+}_k + \boldsymbol{\mathcal{K}}_+(\boldsymbol{z}_{k+1} - \bar{\boldsymbol{z}}_{k+1}), \tag{3.63}$$

$$\boldsymbol{P}^{++}_k = \boldsymbol{P}^{+}_k - \boldsymbol{\mathcal{K}}_+(\boldsymbol{P}^z_{k+1})^{-1}\boldsymbol{\mathcal{K}}^\top_+, \tag{3.64}$$

where the new Kalman gain $\boldsymbol{\mathcal{K}}_+$ and the cross variance $\boldsymbol{P}_{xz+1}$ are given by

$$\boldsymbol{\mathcal{K}}_+ = \boldsymbol{P}_{xz+1}(\boldsymbol{P}^z_{k+1})^{-1}, \tag{3.65}$$

$$\boldsymbol{P}_{xz+1} = \sum_{i=0}^{2L} w^i_c (\boldsymbol{x}^i_k - \bar{\boldsymbol{x}}_k)(\boldsymbol{z}^i_{k+1} - \bar{\boldsymbol{z}}_{k+1})^\top. \tag{3.66}$$

In this way, the distribution $p(\boldsymbol{x}_k \mid \boldsymbol{z}_{0:\zeta})$ can be obtained step by step.

3.4 Summary

In this chapter, the probabilistic estimation methods required to develop the proposed methods in the thesis are presented. GRFs can be used to model the correlation between different points in continuous space, while MRFs represent the correlation in discrete space. Markov chains describe the dynamic behaviours with state change in time, e.g. between free and occupied states. When the Markov chain parameters are unknown and the states are not directly observed, the problem becomes an HMM. The Baum-Welch algorithm used to estimate the HMM parameters is presented. Finally, the Unscented Kalman filter used to estimate the states of a nonlinear dynamical system is presented.

4

Mapping in static environments

This chapter presents three novel mapping methods for static environments. Based on Markov random fields (MRFs) and Gaussian random fields (GRFs), these methods follow a Bayesian viewpoint where a prior distribution is provided as a regularizer. Section 4.1 introduces the classical occupancy grid mapping. Section 4.2 gives the definition of log odds occupancy field and its observation. The first method presented in Section 4.3 is an MRF-based online filter, which focuses on mapping in observed space. In Section 4.4, a prediction method based on MRF is proposed to predict unobserved space. Replacing the MRF by a GRF, the other prediction method is proposed in Section 4.5, which can build maps with high local resolution. The three methods are tested in simulations in the corresponding sections.

4.1 Occupancy grid mapping

Occupancy grid mapping is a classical method in which the map is divided into many grid cells. Each grid cell is regarded as a binary random variable and has two possible states: free and occupied. Normally occupancy probability is used to represent the states, and the map can be represented by a gray scale image shown in Figure 4.1. The darkness of each grid cell corresponds to the likelihood of its occupancy. If one grid cell is certainly free, the darkness is 0. If it is certainly occupied, the darkness is 1.

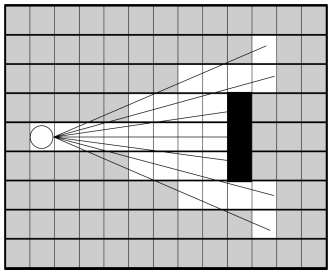


Figure 4.1: An example of a grid map. The black grid cells are occupied, the white ones are free, and the gray ones are unknown.

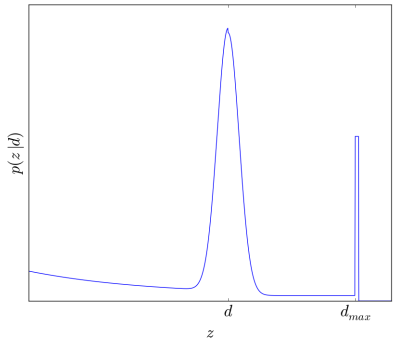


Figure 4.2: Beam model of range finders [TBF05]

Normally range finders, which measure distances to objects, are used to build occupancy grid maps. In this thesis, the beam model of range finders presented in [TBF05] is adopted, where distances are measured along beams as shown in Figure 4.1. This model, shown in Figure 4.2, incorporates four types of measurement errors which are described as follows:

- **Correct range with local measurement noise.** This error is due to the limited resolution of range sensors, atmospheric effect on the measurement signal, and so on. It is usually modelled by a narrow Gaussian distribution in the measurement range $[0, d_{max}]$. Otherwise, the probability density function is 0.
- **Unexpected objects.** In case there are some moving objects in front of the laser sensor, it can only detect the nearest object. The shortest measured range should occur with high probability. It is an exponential distribution in the measurement range $[0, d]$. Otherwise, the probability density function is 0. By analogy with memoryless in time, it is invariant in space. If there is no object until distance d, then from d onwards the distribution is the same.
- **Failures.** Sometimes, range sensors cannot measure distances, for example when laser sensors are pointing at black, light-absorbing objects, or objects in bright light, and if object distance is larger than d_{max}. In these cases, the measured range is always the same as the maximum range d_{max}. It is the probability of the tails of the previous two distributions.
- **Random measurements.** Are characterized by a uniform distribution in the measurement range $[0, d_{max}]$. Otherwise, the probability density function is 0.

In probabilistic form, the beam model is denoted by $p(z \mid d)$, the probability of a measurement range z given the true distance d. In an occupancy grid map, as the one shown in Figure 4.3, the maximum range

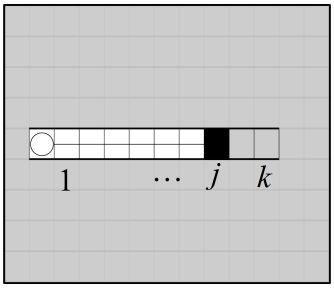


Figure 4.3: A beam in grid map. The length of a beam can be divided into k parts.

d_{max} of a beam can be divided into k grid cells. Assuming the set of these grid cells in the maximum measurement range is denoted by $m' = \{m'_1, \cdots, m'_k\}$ and the true range d corresponds to grid cells $m'_{1:j}$, the beam model of laser sensors in grid maps can be approximated as

$$p(z \mid d) \approx p(z \mid m'_{1:j}). \tag{4.1}$$

The measurement range z does not depend on all the grid cells and only depends on the grid cells covered by the true range d. This is a way of conveying the information that all the space up to the distance d is free, at the distance d is occupied, and unknown after that.

Assuming the set of all the grid cells in a complete map is denoted by $m = \{m_0, m_1, \cdots, m_n\}$ and the observation set is denoted by $z^{1:t} = \{z^1, z^2, \cdots, z^t\}$, the mapping problem aims to estimate the occupancy map given the observations, $p(m \mid z^{1:t})$. Based on the Bayes rule, the estimation of occupancy grid map can be computed recursively by

$$p(m \mid z^{1:t}) = \frac{p(z^t \mid m, z^{1:t-1}) p(m \mid z^{1:t-1})}{p(z^t \mid z^{1:t-1})}, \tag{4.2}$$

where the denominator is a normalizer given by

$$p(z^t \mid z^{1:t-1}) = \sum_m p(z^t \mid m, z^{1:t-1}) p(m \mid z^{1:t-1}), \tag{4.3}$$

and $p(m \mid z^{1:t-1})$ is the previous map estimation given all the past measurements $z^{1:t-1}$. The probability $p(z^t \mid m, z^{1:t-1})$ is the same as the measurement model $p(z \mid m'_{1:j})$ because the measurement range z^t only depends on the true distance. It is not feasible to keep all the joint probabilities of the map $p(m \mid z^{1:t})$ in memory. Instead, marginal distributions can be used to represent the result

$$p(m_i \mid z^{1:t}) = \frac{p(z^t \mid m_i) p(m_i \mid z^{1:t-1})}{\sum_{m_i \in \{occupied, free\}} p(z^t \mid m_i) p(m_i \mid z^{1:t-1})}. \tag{4.4}$$

The probability $p(z^t \mid m_i)$ can be derived from the sensor model $p(z \mid m'_{1:j})$. When the grid cells depend on each other, it is also not feasible to compute $p(z^t \mid m_i)$. Normally the states of different grid cells are assumed to be independent of each other, which is called **state independence** in this thesis. For $m_i \notin m'$, the measurement range z^t does not contribute to it and there is no need to update its estimation. As to $m_i \in m'$, the first step is to compute conditional distributions,

$$p(z^t, m' \setminus m_i \mid m_i) = p(z^t \mid m') p(m' \setminus m_i \mid m_i) \tag{4.5}$$

$$= p(z^t \mid m') p(m' \setminus m_i). \tag{4.6}$$

The subset $m' \setminus m_i$ means the set m' without the grid cell m_i. Because of the state independence, the

subset $m' \setminus m_i$ does not depend on m_i and $p(m' \setminus m_i)$ is the product of all the prior probabilities of the grid cells in the subset $m' \setminus m_i$. Finally, the probability $p(z^t \mid m_i)$ can be obtained from $p(z^t, m' \setminus m_i \mid m_i)$ based on the theory of total probability.

In log odds form [TBF05], the above Bayes filter as equation (4.4) is additive. Assuming the probability that m_i is occupied is denoted by $p(m_i)$ and the free probability is denoted by $p(\bar{m}_i)$, the odds form of the occupancy state of grid cell m_i is the ratio of $p(m_i)$ divided by $p(\bar{m}_i)$,

$$\frac{p(m_i)}{p(\bar{m}_i)}. \tag{4.7}$$

The log odds form of the occupied state is defined by

$$l_i = \log \frac{p(m_i)}{p(\bar{m}_i)}, \tag{4.8}$$

and its range is $(-\infty, \infty)$. The occupancy probability $p(m_i)$ can be obtained by the logistic function

$$p(m_i) = \frac{1}{1 + \exp(-l_i)}. \tag{4.9}$$

By analogy with equation (4.4), the free probability can also be updated recursively as

$$p(\bar{m}_i \mid z^{1:t}) = \frac{p(z^t \mid \bar{m}_i) p(\bar{m}_i \mid z^{1:t-1})}{p(z^t \mid m_i) p(m_i \mid z^{1:t-1}) + p(z^t \mid \bar{m}_i) p(\bar{m}_i \mid z^{1:t-1})}. \tag{4.10}$$

Dividing equation (4.4) by equation (4.10) gives another odds form

$$\frac{p(m_i \mid z^{1:t})}{p(\bar{m}_i \mid z^{1:t})} = \frac{p(z^t \mid m_i)}{p(z^t \mid \bar{m}_i)} \frac{p(m_i \mid z^{1:t-1})}{p(\bar{m}_i \mid z^{1:t-1})}. \tag{4.11}$$

In log odds form, the Bayes filter becomes additive

$$\begin{aligned} \log \frac{p(m_i \mid z^{1:t})}{p(\bar{m}_i \mid z^{1:t})} &= \log \frac{p(z^t \mid m_i)}{p(z^t \mid \bar{m}_i)} + \log \frac{p(m_i \mid z^{1:t-1})}{p(\bar{m}_i \mid z^{1:t-1})} \\ &= \sum_t \log \frac{p(z^t \mid m_i)}{p(z^t \mid \bar{m}_i)} + \log \frac{p(m_i)}{p(\bar{m}_i)}, \end{aligned} \tag{4.12}$$

where $\log \frac{p(z^t|m_i)}{p(z^t|\bar{m}_i)}$ is not in log odds form. Based on the Bayes rule, two posterior state probabilities can be given by

$$p(z^t \mid m_i) = \frac{p(m_i \mid z^t) p(z^t)}{p(m_i)}, \tag{4.13}$$

$$p(z^t \mid \bar{m}_i) = \frac{p(\bar{m}_i \mid z^t) p(z^t)}{p(\bar{m}_i)}. \tag{4.14}$$

Dividing the former by the latter one and taking the log algorithm yields another two log odds forms,

$$\log \frac{p(z^t \mid m_i)}{p(z^t \mid \bar{m}_i)} = \log \frac{p(m_i \mid z^t)}{p(\bar{m}_i \mid z^t)} - \log \frac{p(m_i)}{p(\bar{m}_i)}, \tag{4.15}$$

where $p(m_i \mid z^t)$ is the inverse measurement model [TBF05] that propagates from the measurement to the map. As shown in Figure 4.4, an inverse beam model of laser sensors can be specified by three values:

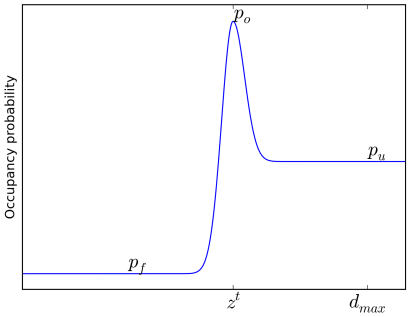


Figure 4.4: Inverse beam model [GAVA08]

- The value p_f represents the occupancy probability that the point is before the end of the measurement range. Its occupancy probability is very low.
- The value p_o represents the occupancy probability that the point is at the end of the measurement range. Its occupancy probability is very high.
- The value p_u represents the occupancy probability that the point is out of the measurement range. Its occupancy probability is the prior $p(m_i)$.

When the measured distance is the maximum range, the occupancy probabilities of all the points in the measurement range are p_f. Given the inverse model, equation (4.12) is rewritten as

$$\log \frac{p(m_i \mid z^{1:t})}{p(\bar{m}_i \mid z^{1:t})} = \log \frac{p(m_i \mid z^t)}{p(\bar{m}_i \mid z^t)} - \log \frac{p(m_i)}{p(\bar{m}_i)} + \log \frac{p(m_i \mid z^{1:t-1})}{p(\bar{m}_i \mid z^{1:t-1})}. \tag{4.16}$$

This equation has clear advantages when compared to equations (4.4) and (4.10). It is an additive equation with a single parameter for each grid cell and avoids truncation problems that arise for probabilities close to 0 or 1 [TBF05].

4.2 Log odds occupancy field

4.2.1 Definition

In classical occupancy grid mapping, the states of different grid cells are independent of each other. Without dependence, occupancy grid maps are typically noisy. Smoothing noisy maps requires a dependence assumption. As mentioned in the previous section, it is not feasible to compute the full joint distribution of the map when dependence is considered because the computational complexity is exponential on the map size. Normally, the occupancy probability is used to represent the built map in occupancy grid mapping and takes a value in range $[0, 1]$. As shown in equation (4.8), the range becomes $(-\infty, +\infty)$ after taking the log odds form. In this section, the log odds form of an occupancy probability is used to represent one point in space and regarded as a random variable l_i defined by equation (4.8). All the random variables in a complete map are denoted by a vector $\boldsymbol{l} = [l_1, \cdots, l_n]^\top$ called log odds occupancy field, where n is the number of random variables. Since the new random variable is a function of the occupancy probability and is not a function of states (occupied, free), the dependence between new random variables and the state independence can hold at the same time.

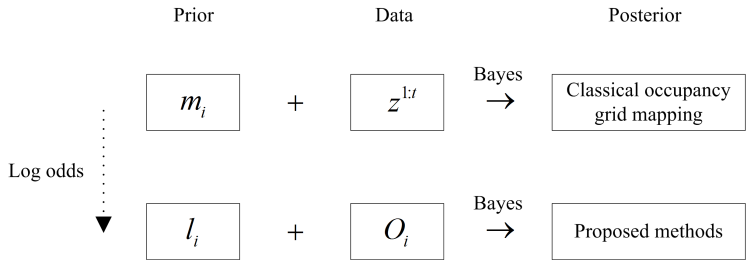


Figure 4.5: Difference between the classical occupancy grid mapping and the proposed methods

4.2.2 Observations

In this chapter, the proposed methods are to estimate the unknown random variables l_i given measurements $z^{1:t}$ following a Bayesian viewpoint where a posterior is obtained given a prior and data. As shown in Figure 4.5, the measurements $z^{1:t}$ are direct observations of states m_i in classical occupancy grid mapping. However, it is difficult to obtain the posterior distribution of l_i directly from the measurements $z^{1:t}$ in proposed methods. For convenience, the direct observation O_i^t of l_i at time t is a function of probabilities $p(z^t \mid m_i)$ and $p(z^t \mid \bar{m}_i)$, defined by

$$O_i^t = \log \frac{p(z^t \mid m_i)}{p(z^t \mid \bar{m}_i)}, \tag{4.17}$$

which is called log observation. Assuming all the measurements $z^{1:t}$ are independent of each other given the map, the computation of the overall observation O_i of l_i is given by

$$\begin{aligned} O_i &= \log \frac{p(z^{1:t} \mid m_i)}{p(z^{1:t} \mid \bar{m}_i)} \\ &= \log \frac{\prod_{j=1}^{t} p(z^j \mid m_i)}{\prod_{j=1}^{t} p(z^j \mid \bar{m}_i)} \\ &= \sum_{j=1}^{t} \log \frac{p(z^j \mid m_i)}{p(z^j \mid \bar{m}_i)} \\ &= \sum_{j=1}^{t} O_i^j. \end{aligned} \tag{4.18}$$

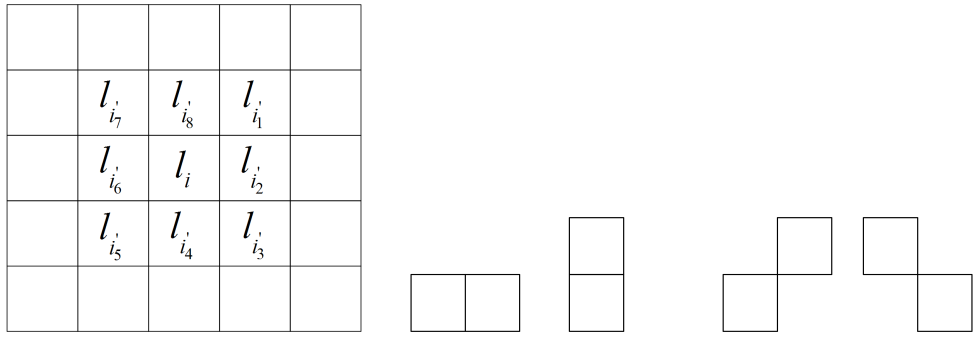
Similarly to equation (4.12), the observations in log odds occupancy field are also additive. However, there is one more term $\log \frac{p(m_i)}{p(\bar{m}_i)}$ in equation (4.12). This term can be regarded as the offset of the observation O_i and called initial log observation denoted by O_i^0, and can be seen as a prior for l_i. The overall observation can be rewritten as the sum

$$O_i = \sum_{j=0}^{t} O_i^j. \tag{4.19}$$

As a consequence, the observation O_i is the result of the Bayes filter in log odds form and called log odds occupancy observation. Based on equation (4.15), the log observation O_i^t can also be obtained from the inverse beam model $p(m_i \mid z^t)$ by

$$O_i^t = \log \frac{p(m_i \mid z^t)}{p(\bar{m}_i \mid z^t)} - O_i^0. \tag{4.20}$$

Similarly to the inverse beam model given in Figure 4.4, the log observation O_i^t for one measurement range can also be specified by three values:



(a) A second-order neighbourhood (b) Pair-variable cliques

Figure 4.6: A second-order neighbourhood and pair-variable cliques

- The free part of the measurement range has a log observation $O_i^t = \log \frac{p_f}{1-p_f} - O_i^0$.
- The occupied point of the measurement range has a log observation $O_i^t = \log \frac{p_o}{1-p_o} - O_i^0$.
- The unobserved part outside the measurement range has a log observation $O_i^t = \log \frac{p_u}{1-p_u} - O_i^0$.

4.3 Filter based on a Markov random field

This section introduces the first proposed method for static environments based on the log odds occupancy field described in the previous section. The map is also divided into grid cells, and every grid cell is represented by a log odds occupancy random variable l_i. The vector $\boldsymbol{l}$ representing all the random variables is regarded as an MRF. A second-order neighbourhood in this MRF is shown in Figure 4.6(a), and every random variable l_i has eight neighbours denoted by $l_{i'}$. As mentioned in Section 3.2.4, there are four kinds of cliques in a second-order neighbourhood. Here, only the pair-variable cliques shown in Figure 4.6(b) in the second-order neighbourhood are considered. For every grid cell, there are 8 pair-variable cliques.

4.3.1 The Markov random field model

Assuming the set of all the pair-variable cliques is denoted by C_2, the prior distribution based on equation (3.37) is

$$p(\boldsymbol{l}) = \frac{1}{Z}\exp\left(-\frac{1}{\mathcal{T}}\sum_{c\in C_2} V_c(\boldsymbol{l})\right), \tag{4.21}$$

where c denotes one clique, the normalizer Z is given by

$$Z = \sum_{\boldsymbol{l}} \exp\left(-\frac{1}{\mathcal{T}}\sum_{c\in C_2} V_c(\boldsymbol{l})\right), \tag{4.22}$$

and $\mathcal{T}$ is the temperature. The clique potential $V_c(l)$ is based on the quadratic deviation between two random variables,

$$V_c(\boldsymbol{l}) = (l_i - l_{i'})^2, \quad (i, i') \in c. \tag{4.23}$$

Since the sum in equation (4.21) is quadratic, the prior distribution can be rewritten as

$$p(\boldsymbol{l}) = \frac{1}{Z}\exp\left(-\frac{2}{\mathcal{T}}\boldsymbol{l}^\top \boldsymbol{\mathcal{A}}\boldsymbol{l}\right), \tag{4.24}$$

where $\boldsymbol{\mathcal{A}}$ is a positive semi-definite $n \times n$ matrix obtained form the cliques and n is the number of grid cells in the map. The maximum of this distribution occurs when $V_c(\boldsymbol{l}) = 0$, which happens when all the random variables take the same value. As a result, there are a lot of maximums in this prior distribution, then $\boldsymbol{\mathcal{A}}$ is singular.

Selecting a grid cell i in the map and expanding all the clique potentials in its second-order neighbourhood gives

$$\sum_{i'}(l_i - l_{i'})^2 = 8l_i^2 + \sum_{i'}(-2l_{i'} + l_{i'}^2). \tag{4.25}$$

This can also be accomplished by selecting the i_{th} row of $\boldsymbol{\mathcal{A}}$ and reshaping it as a squared matrix

$$\begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\ \cdots & 0 & 0 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & -1 & -1 & -1 & 0 & \cdots \\ \cdots & 0 & -1 & 8 & -1 & 0 & \cdots \\ \cdots & 0 & -1 & -1 & -1 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 0 & 0 & \cdots \\ \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

which has the same size as the map. The positions of these nonzero elements in this reshaped matrix are the same as the grid cells in the corresponding neighbourhood. Assuming every grid cell has the same number of neighbours, and $\boldsymbol{\mathcal{A}}$ is a circulant matrix.

The likelihood function is based on the noisy observations O_i whose noise is assumed to be additive Gaussian with mean 0 and variance σ_i^2. If one grid cell is unobserved, the corresponding observation O_i of l_i is given by the equation (4.19) which in this case is just the initial observation O_i^0 and assumed to be 0, and its variance is set to a finite value. Assuming all the observations are independent and denoted by a vector $\boldsymbol{O} = [O_1, \cdots, O_i, \cdots]^\top$, the likelihood function of l is

$$\begin{aligned} p(\boldsymbol{O} \mid \boldsymbol{l}) &= \frac{1}{\prod_i \sqrt{2\pi\sigma_i^2}}\exp\left(-\sum_i \frac{(l_i - O_i)^2}{2\sigma_i^2}\right) \\ &= \frac{1}{\prod_i \sqrt{2\pi\sigma_i^2}}\exp\left(-\frac{1}{2}(\boldsymbol{l} - \boldsymbol{O})^\top \boldsymbol{\Lambda}(\boldsymbol{l} - \boldsymbol{O})\right), \end{aligned} \tag{4.26}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix $\mathrm{diag}(\cdots, 1/\sigma_{i-1}^2, 1/\sigma_i^2, 1/\sigma_{i+1}^2, \cdots)$.

Finally, based on the Bayes rule, the MRF model is obtained

$$p(\boldsymbol{l} \mid \boldsymbol{O}) = \eta p(\boldsymbol{O} \mid \boldsymbol{l})p(\boldsymbol{l}) = \eta \frac{1}{\prod_i \sqrt{2\pi\sigma_i^2}} \frac{1}{Z}\exp\left(-\mathcal{E}(\boldsymbol{l})\right), \tag{4.27}$$

where η is a constant and the posterior energy function $\mathcal{E}(\boldsymbol{l})$ is

$$\mathcal{E}(\boldsymbol{l}) = \frac{2}{\mathcal{T}}\boldsymbol{l}^\top \boldsymbol{\mathcal{A}}\boldsymbol{l} + \frac{1}{2}(\boldsymbol{l} - \boldsymbol{O})^\top \boldsymbol{\Lambda}(\boldsymbol{l} - \boldsymbol{O}). \tag{4.28}$$

4.3.2 Filtering

Both the temperature $\mathcal{T}$ and the variances σ_i^2 can be used to change the weight between the prior knowledge and the likelihood, but they are not independent. Here, the temperature $\mathcal{T}$ is set to 1 and the variances σ_i^2 are modified instead. The mapping result can be obtained by maximizing the posterior distribution $p(\boldsymbol{l} \mid \boldsymbol{O})$, or equivalently, minimizing the posterior energy function $\mathcal{E}(\boldsymbol{l})$. The derivative of $\mathcal{E}(\boldsymbol{l})$ with respect to $\boldsymbol{l}$ is given by

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{l}}\mathcal{E}(\boldsymbol{l}) = 4\boldsymbol{\mathcal{A}}\boldsymbol{l} + \boldsymbol{\Lambda}(\boldsymbol{l} - \boldsymbol{O}). \tag{4.29}$$

Setting this derivative to zero, the linear equation

$$(4\boldsymbol{\mathcal{A}} + \boldsymbol{\Lambda})\boldsymbol{l} = \boldsymbol{\Lambda}\boldsymbol{O} \tag{4.30}$$

is obtained, where $4\boldsymbol{\mathcal{A}} + \boldsymbol{\Lambda}$ is nonsigular. The mapping problem can be solved by

$$\boldsymbol{l} = (4\boldsymbol{\mathcal{A}} + \boldsymbol{\Lambda})^{-1}\boldsymbol{\Lambda}\boldsymbol{O} \tag{4.31}$$

$$= \boldsymbol{\mathcal{H}}^{-1}\boldsymbol{O}, \tag{4.32}$$

where $\boldsymbol{\mathcal{H}} = 4\boldsymbol{\Lambda}^{-1}\boldsymbol{\mathcal{A}} + \boldsymbol{I}$.

If the standard deviations $\sigma_i = \sigma$ are constant for all the grid cells, then $\boldsymbol{\Lambda}^{-1} = \sigma^2\boldsymbol{I}$ and $\boldsymbol{\mathcal{H}}$ becomes a circulant matrix. One row in $\boldsymbol{\mathcal{H}}$ can be reshaped as

$$\begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots \\ \cdots & 0 & 0 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & -4\sigma^2 & -4\sigma^2 & -4\sigma^2 & 0 & \cdots \\ \cdots & 0 & -4\sigma^2 & 1+32\sigma^2 & -4\sigma^2 & 0 & \cdots \\ \cdots & 0 & -4\sigma^2 & -4\sigma^2 & -4\sigma^2 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 0 & 0 & \cdots \\ \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{4.33}$$

As the robot explores the unknown environment, the map size will increase. However, the nonzero elements do not change and only some zeros are added around the border,

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \iddots & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & -4\sigma^2 & -4\sigma^2 & -4\sigma^2 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & -4\sigma^2 & 1+32\sigma^2 & -4\sigma^2 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & -4\sigma^2 & -4\sigma^2 & -4\sigma^2 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \iddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The inverse matrix $\boldsymbol{\mathcal{H}}^{-1}$ is also a circulant matrix. As shown in Figure 4.7, each row in $\boldsymbol{\mathcal{H}}^{-1}$ can also be reshaped as a matrix. Since the neighbourhood is isotropic, the elements in Figure 4.7 are also isotropic

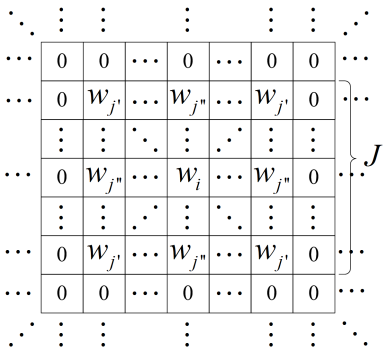


Figure 4.7: One row in $\boldsymbol{\mathcal{H}}^{-1}$ reshaped as a matrix

and sum to 1. If the map size is sufficiently large, the nonzero elements in $\boldsymbol{\mathcal{H}}^{-1}$ become constant, and only some zeros are added beside the outmost elements, as was also the case with $\boldsymbol{\mathcal{H}}$. This means the distant grid cells will not affect the target grid cell. When the nonzero elements in $\boldsymbol{\mathcal{H}}^{-1}$ are known, there is no need to recalculate the inverse matrix as the map size increases. As shown in Figure 4.7, the smallest size of a square that contains these nonzero elements is denoted by J. In order to obtain the J^2 elements, an inverse matrix $\boldsymbol{\mathcal{H}}^{-1}$ corresponding to a map with size $J \times J$ should be computed. The middle row in $\boldsymbol{\mathcal{H}}^{-1}$ containing the J^2 elements is denoted by a vector $\boldsymbol{\mathcal{L}}$. Since the neighbours are isotropic, the size J is an odd number.

When a new observation of a grid cell is obtained, only J^2 grid cells need to be updated. In other words, when a grid cell needs to be estimated, the observations of J^2 grid cells are acquired, which means that this method can be implemented locally. The estimation of l_i at time t is computed by

$$l_i^t = \boldsymbol{\mathcal{L}}[\cdots, O_{i-1}, O_i, \cdots]^\top_{1\times J^2}. \tag{4.34}$$

When a new observation set $\{\cdots, O_{i-1}^{t+1}, O_i^{t+1}, \cdots\}$ is obtained, the estimation can be updated recursively by

$$\begin{aligned} l_i^{t+1} &= \boldsymbol{\mathcal{L}}[\cdots, O_{i-1} + O_{i-1}^{t+1}, O_i + O_i^{t+1}, \cdots]^\top_{1\times \mathcal{J}^2} \\ &= \boldsymbol{\mathcal{L}}[\cdots, O_{i-1}, O_i, \cdots]^\top_{1\times J^2} + \boldsymbol{\mathcal{L}}[\cdots, O_{i-1}^{t+1}, O_i^{t+1}, \cdots]^\top_{1\times J^2} \\ &= l_i^t + \boldsymbol{\mathcal{L}}[\cdots, O_{i-1}^{t+1}, O_i^{t+1}, \cdots]^\top_{1\times J^2}. \end{aligned} \tag{4.35}$$

Underlying this equation, there is another assumption that the variance σ^2 does not change when there are more observations for one grid cell. If the variance changes, the vector $\boldsymbol{\mathcal{L}}$ should be recomputed.

4.3.3 Computational complexity

The filter obtained in the previous section is described as the Algorithm 1 below.

The computational complexity is obtained as follows.

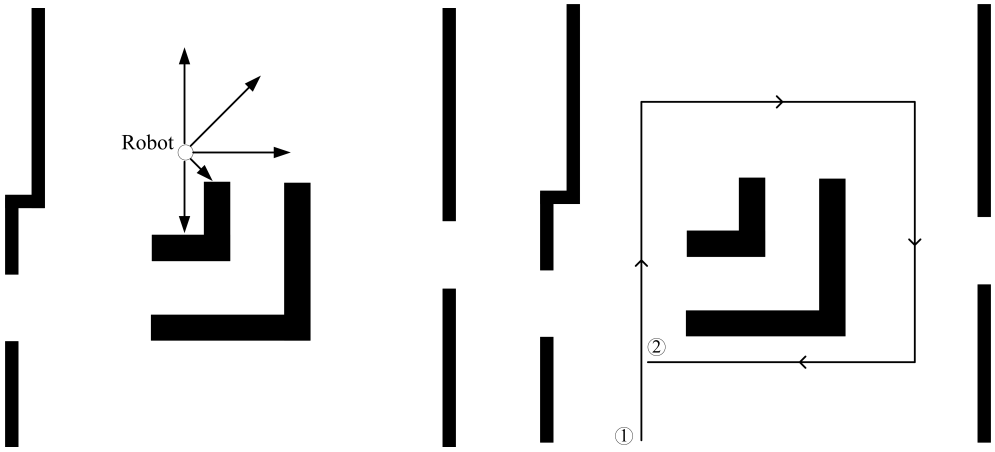
Step 2 After choosing the variance σ^2 and the size J, these values become fixed. Compared with the map size, J is very small and it is very easy to calculate the weight vector $\boldsymbol{\mathcal{L}}$ by inverting a matrix with size $J^2 \times J^2$. The computational complexity of this step is $\mathcal{O}(1)$.

Step 3 The number of new observations depends on the new region observed in a time interval and the size of every grid cell in this region. The grid cells correlated to new observations need to be updated.

Algorithm 1 Filter based on MRF in static environments

Input: New observations $\cdots, O_{i-1}^{t+1}, O_i^{t+1}, \cdots$
Output: l_i^{t+1}

1: Choose the variance σ^2
2: Calculate the weight vector $\boldsymbol{\mathcal{L}}$
3: **for** every grid cell i that is to be updated **do**
4: Obtain l_i^{t+1} using equation (4.35)



(a) Simulated map (b) Trajectory

Figure 4.8: Simulated static map and trajectory. (a): The black parts are walls and the other white space are free. The arrows are measurement directions. (b): The line is the trajectory of the robot and the arrows are moving directions.

The number of these grid cells is denoted by n_c.

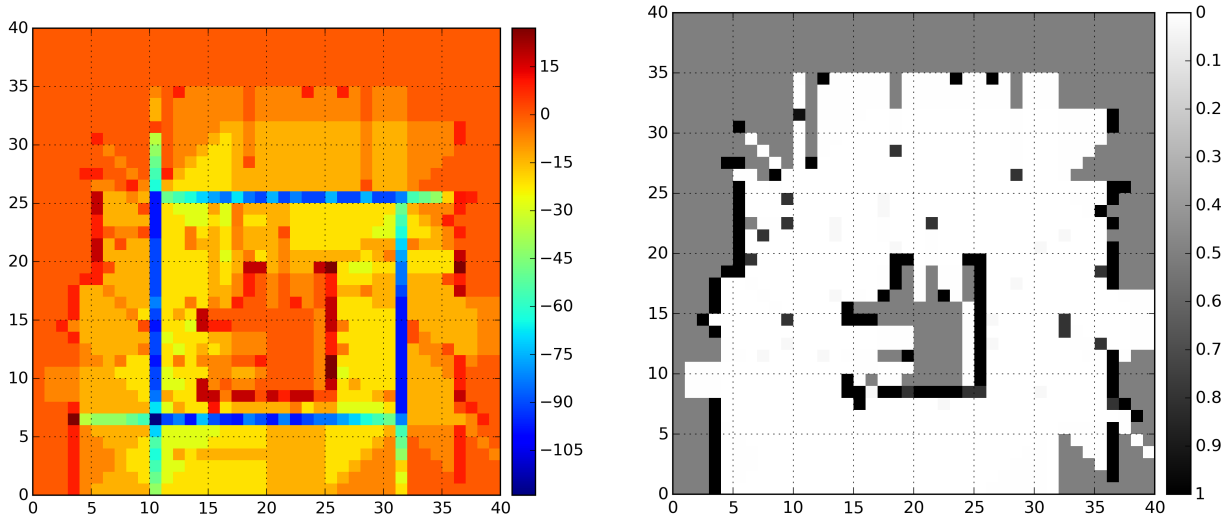
Step 4 Equation (4.35) is very simple and its computational complexity is $\mathcal{O}(1)$.

Finally, since the loop is ran n_c times, the computational complexity of this algorithm is $\mathcal{O}(n_c)$.

4.3.4 Simulation

To illustrate the algorithm, the map shown in Figure 4.8(a) was built, where there are corners and a corridor with open doors. The robot has five measurement directions: 0, $\pm\pi/2$ and $\pm\pi/4$, which are with respect to the robot's reference frame. Each beam has the same width as the grid cells, and the maximum range is set to 9 grid cells. The trajectory is shown in Figure 4.8(b) and the robot runs from ① to ②.

All the initial log observations O_i^0 are set to 0. Following along a line in each one of the measurement directions, the log observations of the free grid cells are set to $O_i^t = -7$. At the end of the measurement range, the grid cell is occupied and its log observation is set to $O_i^t = 9$. If the measurement range is the same as the maximum range, all the grid cells in the measurement range are observed free with log observations set to $O_i^t = -7$. The log observations of the grid cells outside the measurement range are set to $O_i^t = 0$. The log odds occupancy observation of one grid cell is computed by equation (4.17), and the result is shown in Figure 4.9(a). The corresponding occupancy grid map obtained by applying the logistic function is shown in Figure 4.9(b). The free space is observed to have low occupancy probability. The grid



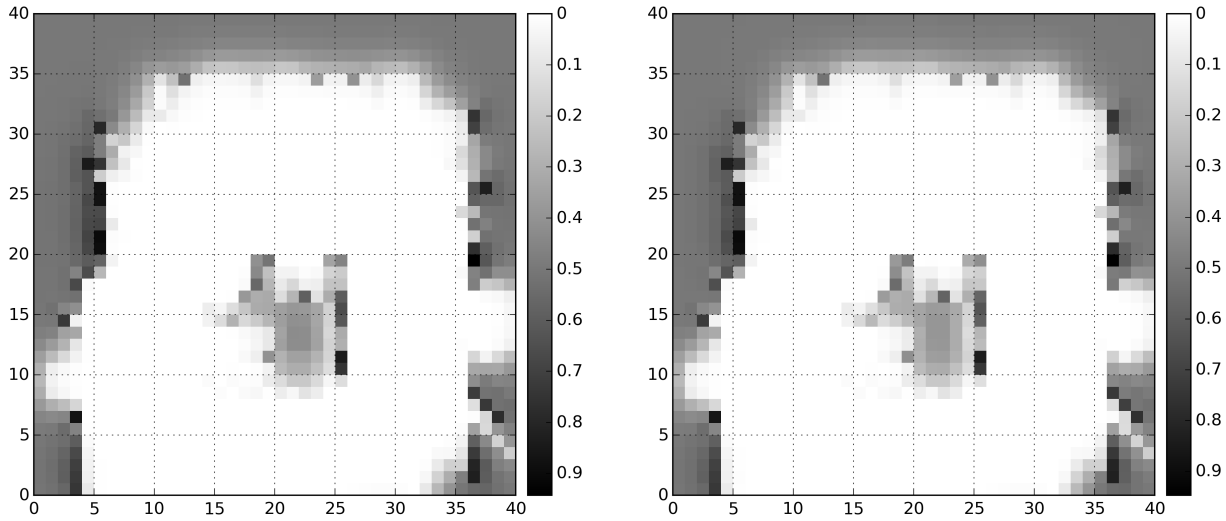
(a) The log odds occupancy observation

(b) The occupancy observation

Figure 4.9: Log odds occupancy observation and occupancy observation of the simulated static map. (a): The colour indicates the occupancy probability in log odds form. (b): The darkness indicates the occupancy probability.

cells along the trajectory are free with the lowest occupancy probabilities. Because of the sensor noise, the walls are not always observed precisely, and some free grid cells can be observed occupied or unknown.

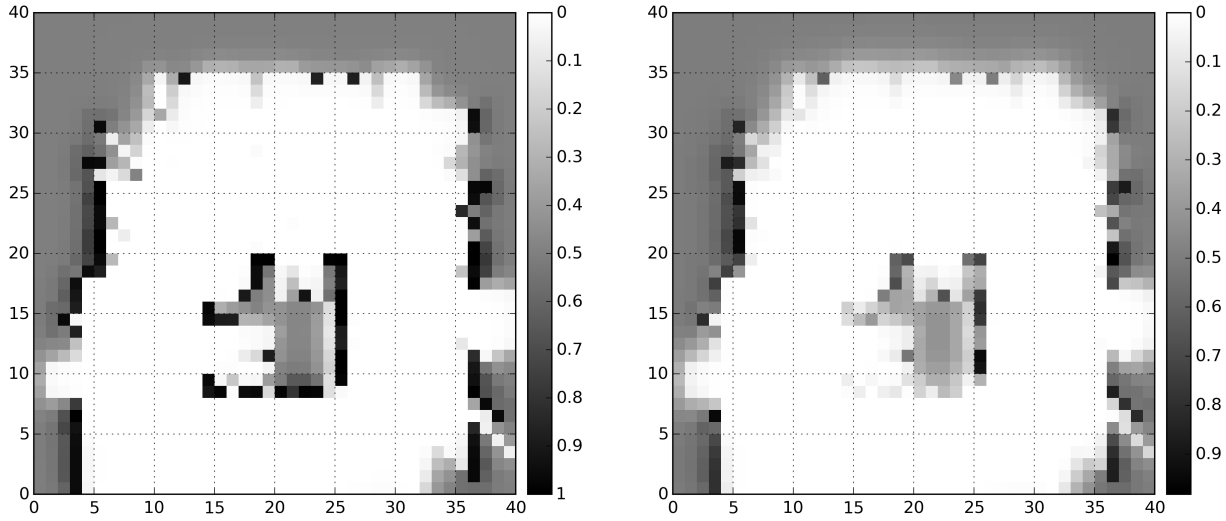
First, the variance σ^2 is set to 0.1, and the size J is chosen. When J is chosen as 11 and 17, the corresponding mapping results are shown in Figure 4.10(a) and 4.10(b), respectively. When J increases from 11 to 17, the difference is negligible. This is a linear filter, and increasing J has little influence on the computational efficiency. The variable J can be set to a large number to ensure accuracy. When J is set to 17, the results of the MRF-based filter with σ^2 0.03 and 0.08 are shown in Figure 4.11. As σ^2 increases, observations become noisier, and the result is smoother.



(a) $J = 11$

(b) $J = 17$

Figure 4.10: Results of MRF-based filter with different J for the simulated static map



(a) $\sigma^2 = 0.03$ (b) $\sigma^2 = 0.08$

Figure 4.11: Results of MRF-based filter with different σ^2 for the simulated static map

4.4 Prediction based on a Markov random field

In the previous section, the variances of unobserved grid cells are finite, and the corresponding observations O_i are set to 0. An unobserved grid cell learns not only from its neighbours but also from this value 0. In this section, the variances of unobserved grid cells are set to infinity. This means one unobserved grid cell can only learn from its neighbours, and this information propagates throughout space. A new method is proposed based on an MRF to deal with this situation.

4.4.1 The Markov random field model

In the MRF model, the map is also divided into grid cells and each grid cell is represented by a random variable l_i. As before, the vector $\boldsymbol{l} = [l_1, \cdots, l_i, \cdots]^\top$ is regarded as an MRF where only the pair-variable cliques in second-order neighbourhoods are considered. The prior distribution is the same as equation (4.24). The observation of l_i is O_i defined by equation (4.17) and all the observations are denoted by a vector $\boldsymbol{O} = [O_1, \cdots, O_i, \cdots]^\top$. If one grid cell is not observed, the variance associated to its observation is infinite. Assuming the index set of observed grid cells are denoted by $\mathcal{I}$ and all the observations are independent, the new likelihood function can be expressed as

$$p(\boldsymbol{O} \mid \boldsymbol{l}) = \frac{1}{\prod_{i\in\mathcal{I}}\sqrt{2\pi\sigma_i^2}} \exp\left(-\sum_{i\in\mathcal{I}} \frac{(l_i - O_i)^2}{2\sigma_i^2}\right), \tag{4.36}$$

where O_i is Gaussian distributed with mean l_i and variance σ_i^2. Since the variances of unobserved grid cells are infinite, their inverses $1/\sigma_i^2$ are set to zeros and their observations do not take any effect on the likelihood function. The variances of observed grid cells are finite. The likelihood function can also be rewritten as

$$p(\boldsymbol{O} \mid \boldsymbol{l}) = \frac{1}{\prod_{i\in\mathcal{I}}\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{1}{2}(\boldsymbol{l} - \boldsymbol{O})^\top \boldsymbol{\Lambda}'(\boldsymbol{l} - \boldsymbol{O})\right), \tag{4.37}$$

where $\boldsymbol{\Lambda}'$ is also a diagonal matrix. If one grid cell is not observed, the corresponding element in $\boldsymbol{\Lambda}'$ is 0. The new MRF model is written as

$$p(\boldsymbol{l} \mid \boldsymbol{O}) = \eta p(\boldsymbol{O} \mid \boldsymbol{l}) p(\boldsymbol{l}) = \eta \frac{1}{\prod_{i \in \mathcal{I}} \sqrt{2\pi\sigma_i^2}} \frac{1}{Z} \exp\left(-\mathcal{E}(\boldsymbol{l})\right), \tag{4.38}$$

where $\mathcal{E}(\boldsymbol{l})$ is rewritten as

$$\mathcal{E}(\boldsymbol{l}) = \frac{2}{\mathcal{T}} \boldsymbol{l}^\top \boldsymbol{\mathcal{A}} \boldsymbol{l} + \frac{1}{2} (\boldsymbol{l} - \boldsymbol{O})^\top \boldsymbol{\Lambda}' (\boldsymbol{l} - \boldsymbol{O}). \tag{4.39}$$

4.4.2 Prediction

The temperature $\mathcal{T}$ is also set to 1. Minimizing the posterior energy function $\mathcal{E}(\boldsymbol{l})$ gives an estimate of $\boldsymbol{l}$. The derivative of $\mathcal{E}(\boldsymbol{l})$ with respect to $\boldsymbol{l}$ is given by

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{l}} \mathcal{E}(\boldsymbol{l}) = 4\boldsymbol{\mathcal{A}} \boldsymbol{l} + \boldsymbol{\Lambda}' (\boldsymbol{l} - \boldsymbol{O}). \tag{4.40}$$

Setting this derivative to zero, the linear equation

$$(4\boldsymbol{\mathcal{A}} + \boldsymbol{\Lambda}') \boldsymbol{l} = \boldsymbol{\Lambda}' \boldsymbol{O} \tag{4.41}$$

is obtained. Without observations, the matrix $4\boldsymbol{\mathcal{A}} + \boldsymbol{\Lambda}'$ is singular and there is no solution to this linear equation. Once there is one observation, the only minimum of the posterior energy function $\mathcal{E}(\boldsymbol{l})$ occurs when all the random variables l_i take the same value as the observation. As a result, the matrix $4\boldsymbol{\mathcal{A}} + \boldsymbol{\Lambda}'$ becomes nonsingular and the prediction equation is written as

$$\boldsymbol{l} = \boldsymbol{\mathcal{H}}'^{-1} \boldsymbol{\Lambda}' \boldsymbol{O}, \tag{4.42}$$

where

$$\boldsymbol{\mathcal{H}}' = 4\boldsymbol{\mathcal{A}} + \boldsymbol{\Lambda}'. \tag{4.43}$$

If one new grid cell is observed and the corresponding variance is σ^2, the matrix $\boldsymbol{\Lambda}'$ can be rewritten as

$$\boldsymbol{\Lambda}' + \mathrm{diag}(\cdots, 0, 1/\sigma^2, 0, \cdots) = \boldsymbol{\Lambda}' + \frac{1}{\sigma^2} \boldsymbol{a}\boldsymbol{a}^\top, \tag{4.44}$$

where $\boldsymbol{a} = [\cdots, 0, 1, 0, \cdots]^\top$. The matrix $\boldsymbol{\mathcal{H}}'$ becomes $\boldsymbol{\mathcal{H}}' + \frac{1}{\sigma^2} \boldsymbol{a}\boldsymbol{a}^\top$. Based on the Sherman–Morrison equation, the inverse matrix can be calculated by

$$(\boldsymbol{\mathcal{H}}' + \frac{1}{\sigma^2} \boldsymbol{a}\boldsymbol{a}^\top)^{-1} = \boldsymbol{\mathcal{H}}'^{-1} - \frac{\boldsymbol{\mathcal{H}}'^{-1} \boldsymbol{a}\boldsymbol{a}^\top \boldsymbol{\mathcal{H}}'^{-1}}{\sigma^2 + \boldsymbol{a}^\top \boldsymbol{\mathcal{H}}'^{-1} \boldsymbol{a}}. \tag{4.45}$$

When more grid cells are observed, this inverse matrix can be computed recursively.

If the map size does not increase, the inverse matrix $\boldsymbol{\mathcal{H}}'^{-1}$ needs to be solved at the beginning, and then it can be computed recursively using equation (4.45). When the map size is very large, it is not easy to compute the inverse matrix of $\boldsymbol{\mathcal{H}}'$. Here, an alternative algorithm is proposed for the case in which the variances of all the observed grid cells are σ^2.

At the beginning, the variances of unobserved grid cells are assumed to be σ^2. In this case, the prediction method in this section is the same as the previous MRF-based filter, and the inverse matrix $\boldsymbol{\mathcal{H}}'^{-1}$ can

be constructed easily by inverting a matrix with a smaller size as described in Section 4.3. In equation 4.44, variances are added into $\boldsymbol{\Lambda}'$ when new grid cells need to be considered. Now, the unobserved grid cells are already considered and their variances should be removed from the matrix $\boldsymbol{\Lambda}'$. When the variance of one unobserved grid cell is removed from $\boldsymbol{\Lambda}'$, the matrix $\boldsymbol{\mathcal{H}}'$ becomes $\boldsymbol{\mathcal{H}}' - \frac{1}{\sigma^2}\boldsymbol{a}\boldsymbol{a}^\top$. Based on the Sherman–Morrison equation, the inverse matrix can be obtained as

$$(\boldsymbol{\mathcal{H}}' - \frac{1}{\sigma^2}\boldsymbol{a}\boldsymbol{a}^\top)^{-1} = \boldsymbol{\mathcal{H}}'^{-1} + \frac{\boldsymbol{\mathcal{H}}'^{-1}\boldsymbol{a}\boldsymbol{a}^\top\boldsymbol{\mathcal{H}}'^{-1}}{\sigma^2 - \boldsymbol{a}^\top\boldsymbol{\mathcal{H}}'^{-1}\boldsymbol{a}}. \tag{4.46}$$

This step should be done again and again until all the unobserved grid cells are removed. Finally, the same prediction can be obtained.

4.4.3 Computational complexity

The prediction based on MRF is described as the Algorithm 2 below.

Algorithm 2 Prediction based on MRF in static environments

Input: New observations $\cdots, O_{i-1}^{t+1}, O_i^{t+1}, \cdots$
Output: $\boldsymbol{l}$

1: Determine σ_i^2
2: Initialize $\boldsymbol{\mathcal{H}}'^{-1}$
3: **for** every new grid cell observed **do**
4: Compute $\boldsymbol{\mathcal{H}}'^{-1}$ using equation (4.45)
5: Obtain $\boldsymbol{l}$ using equation (4.42)

The computational complexity is obtained as follows.

Step 2 In order to initialize the inverse matrix $\boldsymbol{\mathcal{H}}'^{-1}$, at least one grid cell should be observed. At the beginning, the grid cell the robot occupies is assumed to be observed free. Assuming the number of grid cells in the map is n, the computational complexity of this step is $\mathcal{O}(n^3)$.

Step 3 The variances of new observed grid cells should be added into matrix $\boldsymbol{\mathcal{H}}'$. The number of new observed grid cells n_u depends on the new region observed in a time interval and the size of every grid cell in this region. Step 4 should be computed n_u times.

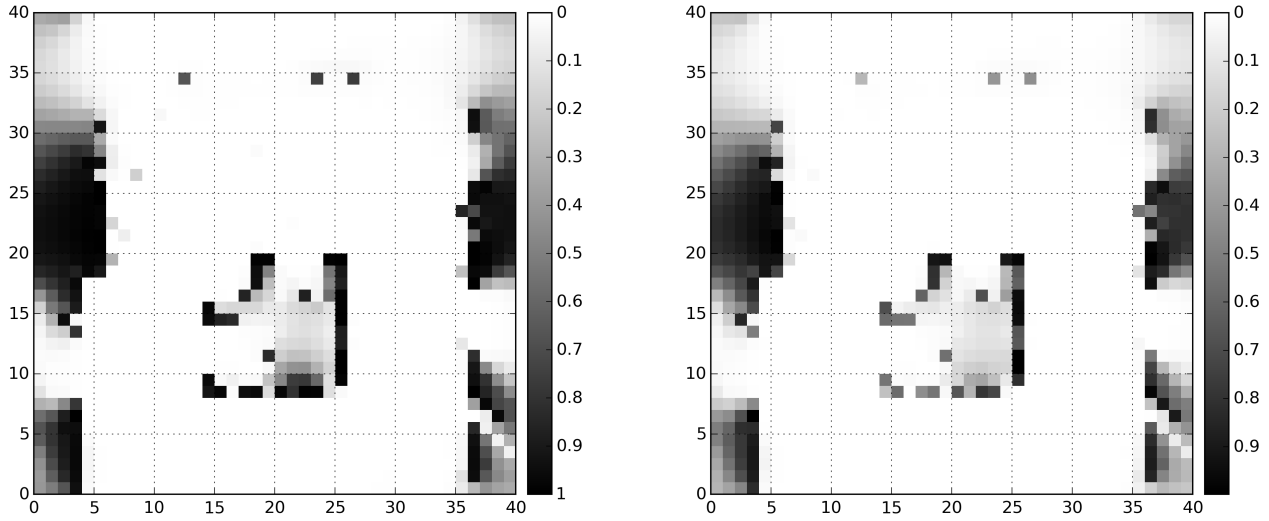
Step 4 Due to the simplicity of vector $\boldsymbol{a}$, the computational complexity of equation (4.45) is $\mathcal{O}(n^2)$.

Step 5 The computational complexity of the final step is just matrix product which is $\mathcal{O}(n^2)$.

Inverting the matrix $\boldsymbol{\mathcal{H}}'$ directly has a computational complexity of $\mathcal{O}(n^3)$. Updating the inverse matrix based on equation (4.45) reduces the computational complexity to $\mathcal{O}(n^2)$.

4.4.4 Simulation

The method proposed in this section is applied to the same map and trajectory used in Section 4.3. The simulated map and trajectory are shown in Figure 4.8, and the observations are shown in Figure 4.9. All the variances are set to σ^2. The results of the MRF-based prediction with different σ^2 are shown in Figure



(a) $\sigma^2 = 0.03$ (b) $\sigma^2 = 0.05$

Figure 4.12: Results of MRF-based prediction with different σ^2 for the simulated static map

4.12. The space behind the left side or right side wall is predicted occupied which in the previous method was predicted unknown (equal probabilities of free and occupied states). The unobserved space on the top is still predicted free. The unobserved space in the corner of the center corridor is predicted free with large uncertainty. Compared with Figure 4.11, the current method is able to propagate the prediction of free space beyond the sensor range limit, as can be seen in the top region of the map. In the observed region of the map, both methods obtain similar results.

4.5 Prediction based on a Gaussian random field

The previous two methods build maps in discrete space. In this section, the log odds occupancy field is regarded as a GRF, and a new method is proposed to build maps in continuous space. In a GRF, some points in observed space should be chosen as training data used to predict any point in continuous space. In the log odds occupancy field, if one point is observed multiple times, its observations are additive. Once the space is now continuous, one question that arises is how to decide if one point is observed multiple times. For convenience, two points close to each other are regarded as the same. As a result, the observed space is also divided into grid cells, and each observed grid cell is represented by its central point. If any point in one grid cell is observed, its central point is assumed to be observed once. After obtaining the training points, maps with arbitrary resolution can be built.

In Section 3.2.1, predictions based on Gaussian processes and GRFs were introduced. In the prediction equation of the Gaussian processes with noisy observations, the inverse of the covariance matrix must be recalculated when new training points arrive. A novel method is proposed to avoid calculating this inverse.

4.5.1 The Gaussian random field model

In the GRF, training points are constrained to the central points of the observed grid cells, and test points can be chosen with arbitrary distances to each other in continuous space. Every point is represented by l_i which is the log odds form of its occupancy probability. The training and test points are represented by a

vector $\boldsymbol{l} = [l_1, \cdots, l_i, \cdots, l_n]^\top$, where n is the number of all the points. The prior distribution is

$$p(\boldsymbol{l}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{K}|}} \exp\left(-\frac{1}{2}(\boldsymbol{l}-\boldsymbol{\mu})^\top \boldsymbol{K}^{-1}(\boldsymbol{l}-\boldsymbol{\mu})\right), \tag{4.47}$$

where $\boldsymbol{\mu} = [\mu_1, \cdots, \mu_i, \cdots, \mu_n]^\top$ is the mean vector and $\boldsymbol{K}$ is the covariance matrix. The covariance function is chosen later. The observation vector of l_i is $\boldsymbol{O} = [O_1, \cdots, O_i, \cdots, O_n]^\top$ where O_i is a log odds observation defined by equation (4.17). The variances of test points are set to infinity. Assuming all the observations in the vector O are independent, the likelihood function $p(\boldsymbol{O} \mid \boldsymbol{l})$ is the same as equation (4.37) where O_i is Gaussian distributed with mean l_i and variance σ_i^2. Test points are not considered in the likelihood function and the corresponding elements in the diagonal matrix Λ' are 0. Based on the Bayes rule, the GRF model is obtained

$$p(\boldsymbol{l} \mid \boldsymbol{O}) = \eta p(\boldsymbol{O} \mid \boldsymbol{l}) p(\boldsymbol{l}) = \eta \frac{1}{\prod_{i\in\mathcal{I}} \sqrt{2\pi\sigma_i^2}} \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{K}|}} \exp\left(-\mathcal{E}(\boldsymbol{l})\right), \tag{4.48}$$

where η is a constant and $\mathcal{E}(\boldsymbol{l})$ is rewritten as

$$\mathcal{E}(\boldsymbol{l}) = \frac{1}{2}(\boldsymbol{l}-\boldsymbol{\mu})^\top \boldsymbol{K}^{-1}(\boldsymbol{l}-\boldsymbol{\mu}) + \frac{1}{2}(\boldsymbol{l}-\boldsymbol{O})^\top \boldsymbol{\Lambda}'(\boldsymbol{l}-\boldsymbol{O}). \tag{4.49}$$

4.5.2 Prediction

Minimizing the posterior energy function $\mathcal{E}(\boldsymbol{l})$ gives an estimate of $\boldsymbol{l}$. The derivative of $\mathcal{E}(\boldsymbol{l})$ with respect to $\boldsymbol{l}$ is written as

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{l}}\mathcal{E}(\boldsymbol{l}) = \boldsymbol{K}^{-1}(\boldsymbol{l}-\boldsymbol{\mu}) + \boldsymbol{\Lambda}'(\boldsymbol{l}-\boldsymbol{O}) \tag{4.50}$$

$$= K^{-1}(\boldsymbol{l}-\boldsymbol{\mu}) + \boldsymbol{\Lambda}'(\boldsymbol{l}-\boldsymbol{\mu}) - \boldsymbol{\Lambda}'(\boldsymbol{O}-\boldsymbol{\mu}). \tag{4.51}$$

Setting this derivative to zero, the linear equation

$$(\boldsymbol{K}^{-1} + \boldsymbol{\Lambda}')(\boldsymbol{l}-\boldsymbol{\mu}) = \boldsymbol{\Lambda}'(\boldsymbol{O}-\boldsymbol{\mu}) \tag{4.52}$$

is obtained, where $\boldsymbol{K}^{-1} + \boldsymbol{\Lambda}'$ is nonsigular. The mapping problem can be solved by

$$\boldsymbol{l} = \boldsymbol{\mu} + (\boldsymbol{K}^{-1} + \boldsymbol{\Lambda}')^{-1}\boldsymbol{\Lambda}'(\boldsymbol{O}-\boldsymbol{\mu}). \tag{4.53}$$

At the beginning, there are no observations and $\boldsymbol{\Lambda}'$ is the zero matrix. If one training point is obtained and the corresponding variance is σ_j^2, the inverse matrix becomes $\boldsymbol{\mathcal{D}} = (\boldsymbol{K}^{-1} + \frac{1}{\sigma_j^2}\boldsymbol{a}\boldsymbol{a}^\top)^{-1}$, where $\boldsymbol{a} = [\cdots, 0, 1, 0, \cdots]^\top$. Based on the Sherman–Morrison equation, the inverse matrix can be written as

$$\boldsymbol{\mathcal{D}} = \boldsymbol{K} - \frac{\boldsymbol{K}\boldsymbol{a}\boldsymbol{a}^\top\boldsymbol{K}}{\sigma_j^2 + \boldsymbol{a}^\top\boldsymbol{K}\boldsymbol{a}}. \tag{4.54}$$

If another training point is obtained and its variance is σ_*^2, the inverse matrix becomes $(\boldsymbol{K}^{-1} + \frac{1}{\sigma_j^2}\boldsymbol{a}\boldsymbol{a}^\top + \frac{1}{\sigma_*^2}\boldsymbol{a}_*\boldsymbol{a}_*^\top)^{-1}$, where $\boldsymbol{a}_* = [\cdots, 0, 1, 0, \cdots]^\top$. Based on Sherman–Morrison equation, this inverse matrix can

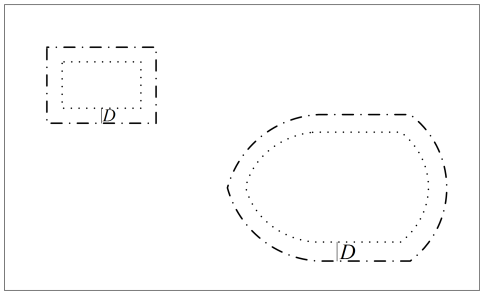


Figure 4.13: Examples of local mapping

be solved by

$$(\boldsymbol{K}^{-1} + \frac{1}{\sigma_j^2} aa^\top + \frac{1}{\sigma_*^2} \boldsymbol{a}_* \boldsymbol{a}_*^\top)^{-1} = \boldsymbol{\mathcal{D}} - \frac{\boldsymbol{\mathcal{D}} \boldsymbol{a}_* \boldsymbol{a}_*^\top \boldsymbol{\mathcal{D}}}{\sigma_*^2 + \boldsymbol{a}_*^\top \boldsymbol{\mathcal{D}} \boldsymbol{a}_*}. \tag{4.55}$$

If there are more observations, it can be done in the same way. As a result, the inverse matrix can be computed without inverting matrices.

Assuming i row in $\boldsymbol{\mathcal{D}}$ is denoted by $\boldsymbol{\mathcal{D}}_i$ and the element at i row and j column is denoted by $\boldsymbol{\mathcal{D}}_{ij}$, every row can be updated by

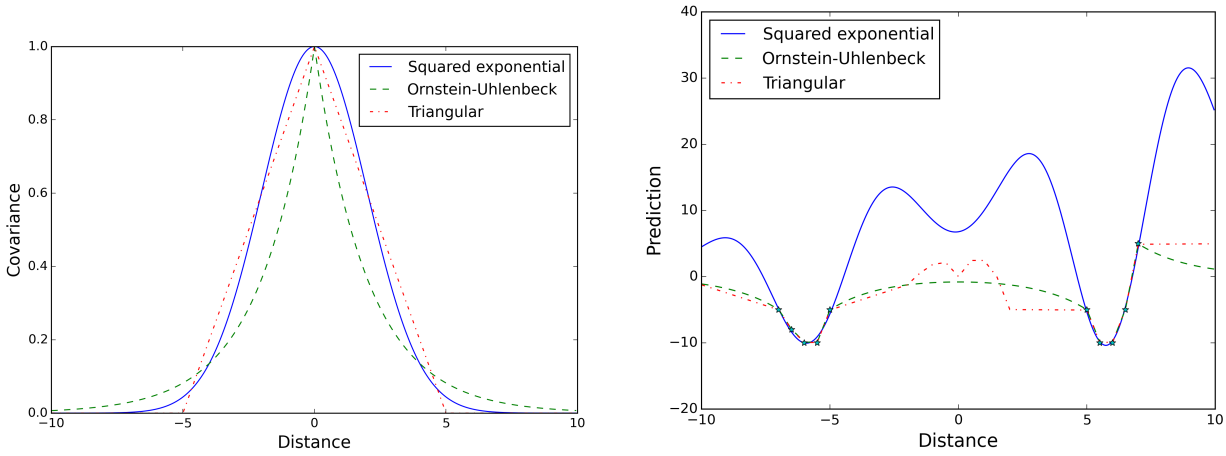
$$\boldsymbol{\mathcal{D}}_i = \boldsymbol{K}_i - \frac{\boldsymbol{K}_{ij}}{\sigma_j^2 + \boldsymbol{K}_{jj}} \boldsymbol{K}_j. \tag{4.56}$$

The covariance function $\boldsymbol{K}$ is a positive semi-definite matrix. When the point with index i is far from the point with index j, the corresponding covariance $\boldsymbol{K}_{ij}$ is close to 0, and there is no need to update the corresponding row $\boldsymbol{\mathcal{D}}_i$. At the beginning, only the elements in $\boldsymbol{K}$ that are greater than a small positive number should be maintained. When predicting one point, only the training points in a small area around it should be considered. The chosen area depends on the covariance function. When a new training point is obtained, only the estimations of the points in a small area need to be updated. This also means there is no need to consider the whole map when only a part of the map is to be predicted. Figure 4.13 shows two examples of local mapping. When the area in dotted line needs to be predicted, the training points in the dash-dot line should be considered. The shape and size of the predicted area can be arbitrary. The augmented distance D depends on the covariance function used.

4.5.3 Choice of the covariance function

How much one point influences its surroundings depends on the covariance function. One point should not correlate to distant points. The squared exponential function, the Ornstein–Uhlenbeck function and the triangular function are possible choices. The corresponding covariance curves with similar influence distances are shown in Figure 4.14(a). When the distance between two points is more than 5, the covariance is 0 or close to 0.

Three covariance functions are tested with the same training data in one dimension, marked by asterisks in Figure 4.14(b). The training points with values less than 0 can be regarded as free points and the ones with values more than 0 are occupied points. All of them have the same variance 0.05. The means of Gaussian processes are set to 0, and the predicted means are shown as three curves. On the left side of the first minimum, the training points are less than 0, the prediction for the squared exponential function exceeds 0. For the squared exponential function and the triangular function, the predictions between two minimums exceed 0. On the right side of the second minimum, there is one occupied training point, and all the prediction curves exceed 0. With some special training points which are less than 0, the prediction for



(a) Covariance curves

(b) Predictions with same dataset

Figure 4.14: Comparison of three covariance functions

the Ornstein–Uhlenbeck function may also exceed 0. Anyway, the Ornstein–Uhlenbeck function provides a qualitatively better prediction than the other two. As a result, the Ornstein–Uhlenbeck function is chosen, and its covariance function is given by

$$\mathcal{C}(c, c') = \sigma_f^2 \exp\left(-\frac{|c - c'|}{\ell}\right), \tag{4.57}$$

where σ_f^2 is the signal variance, the parameter ℓ is the length scale, the variables c and c' are the corresponding coordinates of two random variables.

4.5.4 Computational complexity

The prediction based on GRF is described as the Algorithm 3 below.

Algorithm 3 Prediction based on GRF in static environments

Input: New observations $\cdots, O_{i-1}^{t+1}, O_i^{t+1}, \cdots$
Output: l

1: Determine the training and testing points
2: Initialize covariance matrix $\boldsymbol{K}$
3: Determine σ_i^2
4: **for** every training point **do**
5: **for** every test point correlated to current training point **do**
6: Compute $\boldsymbol{\mathcal{D}}_i$ using equation (4.56)
7: Obtain $\boldsymbol{l}$ using equation (4.53)

The computational complexity is obtained as follows.

Step 1 When new observations are obtained, the first step is to choose the size of the local map as shown in Figure 4.15. Assuming region 1 is one new scan, the local map updated should consist of regions 1 and 2. The observed points in region 3 should be considered when mapping regions 1 and 2. The

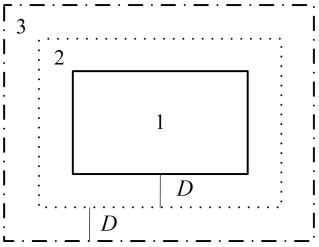


Figure 4.15: Choice of local map

training points are the observed points in regions 1, 2 and 3. The test points can chosen arbitrarily. Because of the noise in observations, the training points in regions 1 and 2 are also chosen as test points. The observed points in regions 1 and 2 are not only training points but also test points.

Step 2 The covariance matrix $\boldsymbol{K}$ is based on the chosen covariance function and its dimension is n. Initializing it has a computational complexity of $\mathcal{O}(n^2)$.

Step 4 The number of training points is denoted by n_o.

Step 5 For every training point, the number of test points correlated with it is not constant and depends on the chosen point in continuous space. Assume the maximum correlated test points is n_m.

Step 6 The computational complexity of this step depends on the number of correlated test and training points. This number is also uncertain and the maximum computational complexity is assumed to be $\mathcal{O}(n_r)$.

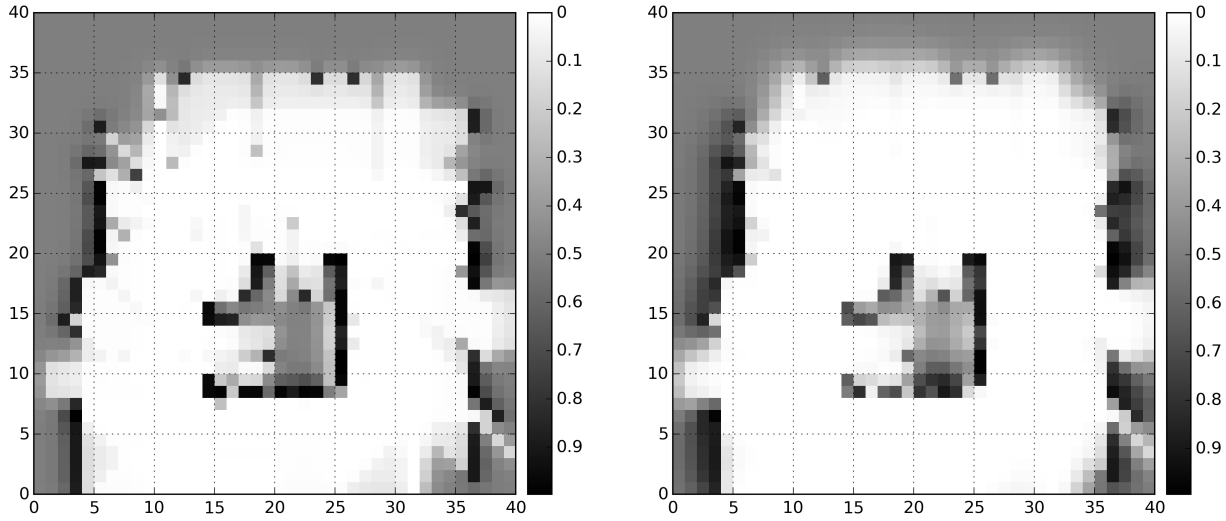
Step 7 Assume the number of the test points is n_t, the computational complexity of the final step is $\mathcal{O}(n_t n_o^2)$.

The computational complexity of the proposed recursive method including steps 4, 5 and 6 is less than $\mathcal{O}(n_o n_m n_r)$. Inverting the matrix has a computational complexity of $\mathcal{O}(n_o^3)$. When n_m and n_r are less than n_o, the proposed recursive method is more advantageous.

4.5.5 Simulation

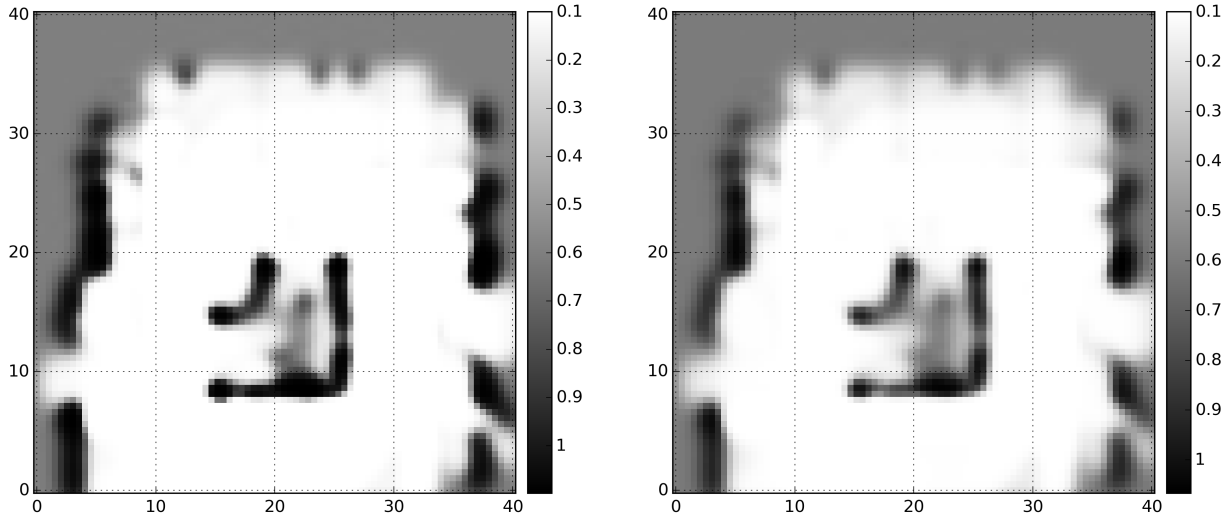
As was done in the other methods based on MRF, the same map and trajectory shown in Figure 4.8 are used. The training points are the central points of observed grid cells in Figure 4.9. The signal variance σ_f^2 is set to 1, and all the variances of observations $\sigma_i^2 = \sigma^2 = 2$. When the length-scale ℓ is chosen as 0.5 and 1, the prediction results are shown in Figure 4.16. Increasing ℓ, more points are correlated, and the map is smoother.

The GRF-based prediction has the ability to build maps with high resolution. In order to improve the resolution, the predicted map with size 40×40 is divided into 80×80 grid cells. The training data is the same as the previous simulation. The whole map is also divided into 8×8 local maps. The global map is built tiling the space with the local maps. The signal variance σ_f^2 and the length-scale ℓ are set to 1. The results with different σ^2 are shown in Figure 4.17. With the same σ^2, the prediction results in Figure 4.16(b) and 4.17(a) are similar. The main difference between them is their resolutions. The smoothness of the global map can be ensured by overlapping observations.



(a) $\ell = 0.5$ (b) $\ell = 1$

Figure 4.16: Results of GRF-based prediction with different ℓ for the simulated static map



(a) $\sigma^2 = 2$ (b) $\sigma^2 = 5$

Figure 4.17: Results of MRF-based prediction with different σ^2 for the simulated static map

4.6 Comparison

The MRF can be divided in two parts, a filter and a prediction. The MRF-based filter is an online method, which can filter out noisy measurements while building a map. The MRF-based prediction can be used to predict the unobserved space. When the whole map is observed, both methods are equivalent. Otherwise, when there is unobserved space, the two methods are not equivalent, but the difference is small in the observed space. In GRFs, one point only influences a certain small area. The GRF-based prediction works in continuous space and can be implemented locally without losing smoothness. It also allows the map to be built with high resolution.

Choosing Figures 4.9(b), 4.11(a), 4.12(a) and 4.17(a) as the representative results of the classical occupancy

grid mapping (OGM) methods and the three proposed methods, the quantitative comparison between different methods is shown as Table 4.1. The occupancy probability between 0.47 and 0.53 is classified as unknown. While the occupancy probability more than 0.53 is classified as occupied and the occupancy probability less than 0.47 is classified as free. The proposed methods predict more true free (TF) and false occupied (FO) space, because the space behind the wall is predicted occupied. The MRF-based filter and the GRF-based prediction produce similar results. However, the latter can build continuous maps with any resolution. The MRF-based prediction can predict more space than the other two.

Table 4.1: Comparison between results for the simulated static map. TF= True free, TO= True occupied, FF= False free, FO= False occupied, UN=Unknown.

Free grid cell	TF	TO	FO	FF	UN
Classical OGM (Figure 4.9(b))	897	75	27	45	556
MRF-based filter (Figure 4.11(a))	1046	79	85	66	324
MRF-based prediction (Figure 4.12(a))	1296	77	137	73	17
GRF-based prediction (Figure 4.17(a))	1019	91	76	55	359

The methods proposed in this chapter are considered occupancy mapping methods. Compared with the classical occupancy grid mapping shown in Figure 4.9(b), the MRF-based filter can produce a smooth map while the MRF-based prediction is able to predict unobserved space. However, the computation in the latter is slightly more complex. The GRF-based prediction can also predict the unobserved space. Even though it is not so computational efficient as the classical occupancy grid mapping, it can also be implemented online. In the Gaussian process occupancy map (GPOM) mentioned in Section 2.3.2, it is not considered that one point is observed multiple times. In the proposed methods, the point observed multiple times have low uncertainty.

4.7 Summary

In this chapter, different mapping methods are proposed for static environments. The occupancy probability in log odds form is applied to represent the map that is regarded as a random field. An MRF-based filter is proposed under the assumption that the variance of unobserved space is not infinity. When all the variances are the same, the filter becomes very computationally efficient and can be implemented online. An MRF-based prediction method is proposed when the unobserved space is not considered in the likelihood function. Given the prior, the unobserved space can be predicted and the result is obtained by solving an inverse matrix. Based on the Sherman–Morrison equation, the inverse matrix is computed recursively when new grid cells are observed. Finally, A GRF-based prediction is proposed. Given a Gaussian prior, the prediction can be done locally. Instead of the prediction equation with an inverting matrix problem, a new recursive method is proposed to reduce the computational complexity. The GRF-based prediction can build maps with arbitrary resolution. These methods are tested by the same simulated data.

5

Mapping in dynamic environments

The previous chapter presents novel mapping methods for static environments. This chapter concerns the mapping problem in dynamic environments where robots have to interact with moving objects. A different model is used to tackle the dynamic objects, and two methods are proposed using Markov random fields (MRFs) and Gaussian random fields (GRFs). Section 5.1 introduces hidden Markov models (HMMs) for dynamic environments. Based on MRFs, the first method is proposed in Section 5.2. The other method based on GRFs is presented in Section 5.3. The proposed methods are also tested in simulations in the corresponding sections.

5.1 Hidden Markov models for dynamic environments

5.1.1 The hidden Markov model

The main difference between static and dynamic environments is the existence of unpredictable dynamic objects. In dynamic environments, one point c in space may be occupied or free in different time instants. Here, its next state m_c^{t+1} is assumed to only depend on the current one m_c^t and a Markov chain is applied

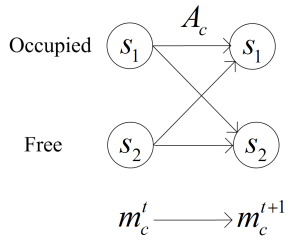


Figure 5.1: Markov chain for dynamic environments

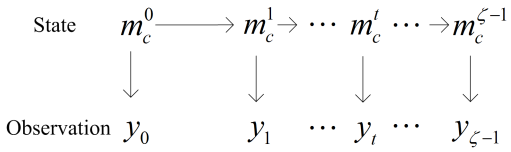


Figure 5.2: HMM for dynamic environments

to model the dynamic behaviour. The occupied and free states are denoted by s_1 and s_2, respectively. The Markov chain is shown in Figure 5.1, where the probability transition matrix for m_c is denoted by $\boldsymbol{A}_c = \{a_{ij}^c\}$ and assumed to be time-invariant.

Defining a grid cell whose central point is c, this point is measured once if one measurement beam z passes by this grid cell. The Markov chain is a discrete model in time. Between two time instants, the state is assumed to be constant and may be measured multiple times. The measurement beams for the current state m_c^t are denoted by z_i^t with the same superscript t and different subscripts, and the measurement sequence is denoted by $y_t = (z_1^t, z_2^t, \cdots)$. Because of sensor noise, the states cannot be observed certainly, and an HMM can be applied. The graphical model of an HMM is shown in Figure 5.2, where ζ is the number of the measurement sequences. The corresponding emission probabilities are $p(y_t \mid m_c^t)$. Since robots always moves in space, the observation sequences for the states in different time instants may be different, and the emission probabilities are also different. However, they are not unknown parameters. Assuming independent observations, the emission probabilities can be derived by

$$p(y_t \mid m_c^t) = \prod_i p(z_i^t \mid m_c^t), \tag{5.1}$$

where $p(z_i^t \mid m_c^t)$ is the probability of the measurement z_i^t given the state m_c^t. How to obtain $p(z_i^t \mid m_c^t)$ from the sensor model is already mentioned in Section 4.1.

Besides two transition probabilities, the overall expected duration mentioned in Section 3.2.2 provides an alternative way to analyse the dynamic behaviour. The state distribution (occupied, free) of one point may change with time and the overall expected durations at different times may be different. For convenience, the overall expected duration at a stationary state is applied and given by

$$E(d) = \mathcal{G}_1 \frac{1}{1 - a_{11}^c} + \mathcal{G}_2 \frac{1}{1 - a_{22}^c}, \tag{5.2}$$

where $\mathcal{G}_1$ and $\mathcal{G}_2$ are occupancy and free probabilities of the stationary distribution, respectively. The

stationary distribution can be obtained by solving

$$\begin{cases} [\mathcal{G}_1\ \mathcal{G}_2] \begin{bmatrix} a_{11}^c & 1-a_{11}^c \\ 1-a_{22}^c & a_{22}^c \end{bmatrix} = [\mathcal{G}_1\ \mathcal{G}_2], \\ \mathcal{G}_1 + \mathcal{G}_2 = 1. \end{cases} \tag{5.3}$$

As a result, the state probabilities $\mathcal{G}_1$ and $\mathcal{G}_2$ are given by

$$\mathcal{G}_1 = \frac{1-a_{22}^c}{2-a_{11}^c-a_{22}^c}, \tag{5.4}$$

$$\mathcal{G}_2 = \frac{1-a_{11}^c}{2-a_{11}^c-a_{22}^c}. \tag{5.5}$$

Given two transition probabilities a_{11}^c and a_{22}^c, the overall expected duration can be computed to indicate how dynamic one point is. High dynamic points will have short overall expected durations.

5.1.2 Parameter estimation

As mentioned in Section 3.2.3, it is not possible to estimate transition probabilities a_{11}^c and a_{22}^c by maximizing likelihood directly. The expectation maximization (EM) algorithm can be applied to estimate the transition probabilities and the initial state probabilities. The parameters are denoted by $\theta_c = \{\rho_i^c, a_{ij}^c\}$, where ρ_i^c represents the initial state probability $p(m_c^0 = s_i)$. Assuming an observation sequence is denoted by $O = (y_0, y_1, \cdots, y_{\zeta-1})$ and an underlying state sequence is denoted by $\mathcal{M}_c = (m_c^0, m_c^1, \cdots, m_c^{\zeta-1})$, the likelihood function of θ_c given the observation sequence and the underlying state sequence is

$$p(O, \mathcal{M}_c \mid \theta_c) = p(m_c^0)p(y_0 \mid m_c^0) \prod_{t=1}^{\zeta-1} p(m_c^t \mid m_c^{t-1})p(y_t \mid m_c^t). \tag{5.6}$$

However, an observation sequence y_t does not include all of the space. When m_c^t is not observed, the emission probabilities $p(y_t \mid m_c^t)$ are set to 1. In order to estimate the parameters, the EM algorithm is applied to recursively maximize a Q function given by

$$Q(\theta_c, \theta_c^{(k)}) = \sum_{\mathcal{M}_c} p(\mathcal{M}_c \mid O, \theta_c^{(k)}) \log p(\mathcal{M}_c, O \mid \theta_c) \tag{5.7}$$

$$= \sum_{\mathcal{M}_c} p(\mathcal{M}_c \mid O, \theta_c^{(k)}) \left(\log p(O \mid \mathcal{M}_c, \theta_c) + \log p(\mathcal{M}_c \mid \theta_c)\right) \tag{5.8}$$

$$= \sum_{\mathcal{M}_c} p(\mathcal{M}_c \mid O, \theta_c^{(k)}) \log p(O \mid \mathcal{M}_c, \theta_c) + \sum_{\mathcal{M}_c} p(\mathcal{M}_c \mid O, \theta_c^{(k)}) \log p(\mathcal{M}_c \mid \theta_c), \tag{5.9}$$

where the sum is over all the possible state sequences $\mathcal{M}_c$, and $\theta_c^{(k)}$ represents the parameters obtained in iteration k. Given the state sequence $\mathcal{M}_c$, the observation sequence O is conditionally independent of the parameters θ_c. As a result, the probabilities $p(O \mid \mathcal{M}_c, \theta_c)$ can be rewritten as $p(O \mid \mathcal{M}_c)$ which can be derived from the emission probabilities $p(y_t \mid m_c^t)$. As a consequence, the first term in equation (5.9) is constant. The parameters can be equivalently obtained by maximizing the second term. Similarly to

equation (3.25), the second term is rewritten as

$$\begin{aligned}\sum_{\mathcal{M}_c} p(\mathcal{M}_c \mid O, \theta_c^{(k)}) \log p(\mathcal{M}_c \mid \theta_c) &= \sum_{i=1}^{2} \gamma_i^c(0) \log \rho_i^c + \sum_{t=1}^{\zeta-1} \sum_{i=1}^{2} \sum_{j=1}^{2} \xi_{ij}^c(t) \log a_{ij}^c \\ &= \sum_{i=1}^{2} \gamma_i^c(0) \log \rho_i^c + \sum_{t=1}^{\zeta-1} \sum_{i=1}^{2} \xi_{1i}^c(t) \log a_{1i}^c + \sum_{t=1}^{\zeta-1} \sum_{i=1}^{2} \xi_{2i}^c(t) \log a_{2i}^c,\end{aligned} \tag{5.10}$$

where $\gamma_i^c(t)$ represents $p(m_c^t = s_i \mid O, \theta_c^{(k)})$ which is the probability of being in state s_i in time instant t given the observation sequence O and the parameters $\theta_c^{(k)}$, and $\xi_{ij}^c(t)$ represents $p(m_c^t = s_i, m_c^{t+1} = s_j \mid O, \theta_c^{(k)})$ which is the probability of being in state s_i in time instant t and s_j in time instant $t+1$ given the observation sequence O and parameters $\theta_c^{(k)}$. Since the three terms in equation (5.10) contain different parameters ρ_1^c, a_{11}^c, and a_{22}^c, they can be maximized individually. The three terms can be represented by the functions $f(\rho_1^c)$, $f(a_{11}^c)$ and $f(a_{22}^c)$, defined by

$$\begin{aligned} f(\rho_1^c) &= \sum_{i=1}^{2} \gamma_i^c(0) \log \rho_i^c \\ &= \gamma_1^c(0) \log \rho_1^c + \gamma_2^c(0) \log (1 - \rho_1^c), \end{aligned} \tag{5.11}$$

$$\begin{aligned} f(a_{11}^c) &= \sum_{t=1}^{\zeta-1} \sum_{i=1}^{2} \xi_{1i}^c(t) \log a_{1i}^c \\ &= \sum_{t=1}^{\zeta-1} \xi_{11}^c(t) \log a_{11}^c + \sum_{t=1}^{\zeta-1} \xi_{12}^c(t) \log (1 - a_{11}^c), \end{aligned} \tag{5.12}$$

$$\begin{aligned} f(a_{22}^c) &= \sum_{t=1}^{\zeta-1} \sum_{i=1}^{2} \xi_{2i}^c(t) \log a_{2i}^c \\ &= \sum_{t=1}^{\zeta-1} \xi_{21}^c(t) \log (1 - a_{22}^c) + \sum_{t=1}^{\zeta-1} \xi_{22}^c(t) \log a_{22}^c. \end{aligned} \tag{5.13}$$

Maximizing the first function $f(\rho_1^c)$ gives the estimation of the initial probabilities,

$$\rho_i^{c(k+1)} = \gamma_i^c(0). \tag{5.14}$$

Maximizing the other two functions $f(a_{22}^c)$ gives the estimation of the transition probabilities,

$$a_{ii}^{c(k+1)} = \frac{\sum_{t=1}^{\zeta-1} \xi_{ii}^c(t)}{\sum_{t=1}^{\zeta-1} \gamma_i^c(t)}. \tag{5.15}$$

Computing the probabilities $\gamma_i^c(t)$ and $\xi_{ij}^c(t)$ requires temporary variables $\alpha_i^c(t)$ and $\beta_i^c(t)$. The variable $\alpha_i^c(t)$ denotes $p(y_0, y_1, \cdots, y_t, m_c^t = s_i \mid \theta_c^{(k)})$, which is the probability of seeing the observation sequence $y_0, y_1, \cdots, y_t$ and being in state s_i. The variable $\beta_i(t)$ denotes $p(y_{t+1}, \cdots, y_{\zeta-1} \mid m_c^t = s_i, \theta_c^{(k)})$, which is the probability of the observation sequence $y_{t+1}, \cdots, y_{\zeta-1}$ given starting state s_i. Similarly to the

forward-backward algorithm used in Section 3.2.3, the probabilities $\gamma_i^c(t)$ and $\xi_{ij}^c(t)$ can be computed by

$$\gamma_i^c(t) = \frac{\alpha_i^c(t)\beta_i^c(t)}{\sum_{j=1}^{2}\alpha_j^c(t)\beta_j^c(t)}, \tag{5.16}$$

$$\xi_{ij}^c(t) = \frac{\alpha_i^c(t)a_{ij}\beta_j^c(t+1)p(y^{t+1} \mid m_c^{t+1} = s_j)}{\sum_{i=1}^{2}\sum_{j=1}^{2}\alpha_i^c(t)a_{ij}\beta_j^c(t+1)p(y^{t+1} \mid m_c^{t+1} = s_j)}. \tag{5.17}$$

In one time instant, the map is assumed to be static, and occupancy grid mapping can be applied to build a temporal occupancy map. Because the transition matrix is unknown, the state probabilities $p(m_c^t = s_1)$ and $p(m_c^t = s_2)$ are also unknown. All the state probabilities are temporarily set to 0.5 and the posterior state probabilities $p(m_c^t = s_1 \mid y_t)$ and $p(m_c^t = s_2 \mid y_t)$ can be obtained. However, the emission probabilities $p(y_t \mid m_c^t = s_1)$ and $p(y_t \mid m_c^t = s_2)$ are required for HMM. Based on the Bayes rule, the posterior probability of the occupied state is given by

$$p(m_c^t = s_1 \mid y_t) = \frac{p(y_t \mid m_c^t = s_1)p(m_c^t = s_1)}{p(y_t)}, \tag{5.18}$$

where $p(y_t)$ is a constant and $p(m_c^t = s_1)$ is the occupancy probability. Similarly, the posterior probability of the free state is given by

$$p(m_c^t = s_2 \mid y_t) = \frac{p(y_t \mid m_c^t = s_2)p(m_c^t = s_2)}{p(y_t)}, \tag{5.19}$$

where $p(m_c^t = s_2)$ is the free probability. Because the prior probabilities $p(m_c^t)$ are set to 0.5, the posterior state probabilities $p(m_c^t = s_1 \mid y_t)$ and $p(m_c^t = s_2 \mid y_t)$ are the normalized version of $p(y^t \mid m_c^t = s_1)$ and $p(y^t \mid m_c^t = s_2)$. Replacing $p(y^t \mid m_c^t)$ by $p(m_c^t \mid y_t)$ to compute $\alpha_i^c(t)$ and $\beta_i^c(t)$ gives temporary variables

$$\hat{\alpha}_i^c(t) = \eta_\alpha^c(t)\alpha_i^c(t), \tag{5.20}$$

$$\hat{\beta}_i^c(t) = \eta_\beta^c(t)\beta_i^c(t), \tag{5.21}$$

where $\eta_\alpha^c(t)$ and $\eta_\beta^c(t)$ are constants. Using these new variables directly to calculate $\gamma_i^c(t)$ and $\xi_{ij}^c(t)$ by equations (5.16) and (5.17), the same $\gamma_i^c(t)$ and $\xi_{ij}^c(t)$ can be obtained,

$$\gamma_i^c(t) = \frac{\hat{\alpha}_i^c(t)\hat{\beta}_i^c(t)}{\sum_{j=1}^{2}\hat{\alpha}_j^c(t)\hat{\beta}_j^c(t)}, \tag{5.22}$$

$$\xi_{ij}^c(t) = \frac{\hat{\alpha}_i^c(t)a_{ij}^c\hat{\beta}_j^c(t+1)p(m_c^{t+1} = s_j \mid y^{t+1})}{\sum_{i=1}^{2}\sum_{j=1}^{2}\hat{\alpha}_i^c(t)a_{ij}^c\hat{\beta}_j^c(t+1)p(m_c^{t+1} = s_j \mid y^{t+1})}. \tag{5.23}$$

Finally these constants are cancelled. As a result, the posterior state probabilities $p(m_c^t \mid y_t)$ can be used to estimate the parameters conveniently instead of the emission probabilities $p(y_t \mid m_c^t)$. For the case that one point is not observed in one time instant, setting the corresponding emission probabilities $p(y_t \mid m_c^t)$ to 0.5 also gives the same result.

5.1.3 Computational complexity

The EM method is described as the Algorithm 4 below.

Algorithm 4 Expectation maximization

Input: Measurements z_i^t
Output: $\boldsymbol{a}_1^{(k)}$ and $\boldsymbol{a}_2^{(k)}$
1: Initialize $\rho_i^{c(0)}$, $a_{ii}^{c(0)}$
2: Build occupancy grid maps with temporary state probabilities 0.5 and obtain $p(m_c^t \mid y_t)$
3: **for** every observed point c **do**
4: $k = 0$
5: **while** $k <$ maximum iteration **do**
6: Calculate $\alpha_i^c(t)$ and $\beta_i^c(t)$ using $p(m_c^t \mid y_t)$
7: Calculate $\gamma_i^c(0)$ and $\xi_{ij}^c(t)$ using equations (5.22) and (5.23)
8: Update the initial probabilities $\rho_i^{c(k+1)}$ using equation (5.14)
9: Update the transition probabilities $a_{ii}^{c(k+1)}$ using equation (5.15)
10: $k = k + 1$

The computational complexity is obtained as follows.

Step 2 The observed space can be divided into grid cells, and each grid cell is represented by its central point. An occupancy grid map is built for every time instant. For one point in one time instant, its state probabilities can be iteratively updated by equation (4.4). The iteration number is the same as the number of measurements which is limited in one time instant. As a result, the computational complexity of calculating the state probabilities for one point in one time instant is $\mathcal{O}(1)$, and the computational complexity of this step is $\mathcal{O}(\zeta n_o)$, where ζ is the number of the measurement sequences and n_o is the number of observed points.

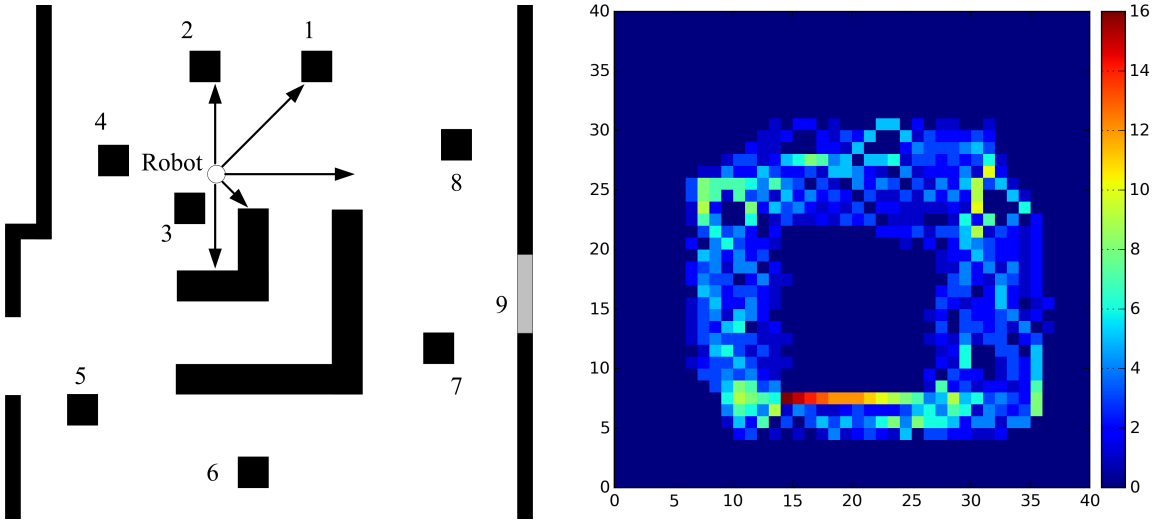
Step 5 Different points have different convergence speeds, and the detail is discussed later in Section 5.1.5.

Steps 6 and 7 For every point, the forward-backward procedure has a computational complexity of $\mathcal{O}(\zeta)$.

Steps 8 and 9 For every point, the update of parameters has a computational complexity of $\mathcal{O}(1)$.

5.1.4 Simulation

Based on the simulated static environment shown in Figure 4.8(a), eight dynamic objects and a dynamic door are added to simulate the dynamic environment shown in Figure 5.3(a). These dynamic objects change their states with different frequencies. The corresponding state durations of eight dynamic objects and the dynamic door are $\Delta t/3$, $2\Delta t/3$, Δt, $2\Delta t$, $4\Delta t$, $13\Delta t$, $26\Delta t$, $50\Delta t$, and $60\Delta t$, where Δt denotes the time between two time instants. After one state duration, the corresponding object may change its state or keep its state with the same probability 0.5. Dynamic objects 1 and 2 may change their states during one Δt and they are used to show if very high dynamic objects can be modelled correctly. The map is divided into grid cells, and the speed of the robot is 3 grid cells per Δt. The robot has five measurement directions: 0, $\pm\pi/2$ and $\pm\pi/4$, which are relative to the robot's orientation. The maximum range is 9 grid cells. After the robot runs around the middle corner clockwise 20 loops, the trajectory is shown in Figure 5.3(b). The values are the number of times the robot passes by each grid cell. Every side of some dynamic objects can be observed. Meanwhile, the borders of some dynamic objects opposite to the trajectory cannot be observed due to the obstacles in the middle of the room and distances being larger than the measurement range.



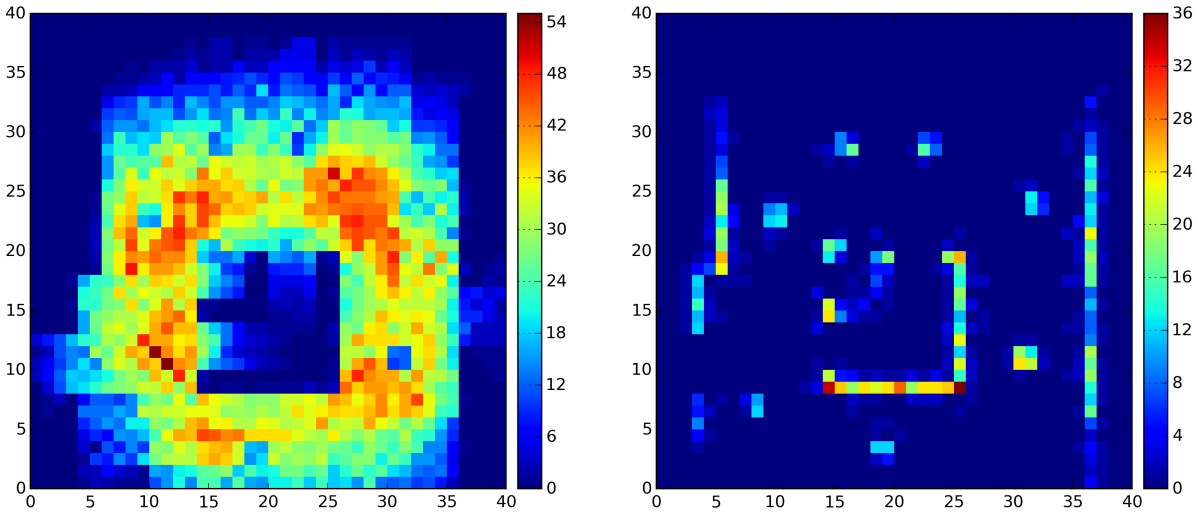
(a) Simulated dynamic environment

(b) Simulated path of the robot

Figure 5.3: Simulated dynamic environment and path. (a): The gray part is a dynamic door and the black squares are dynamic objects. The other black parts are static walls. The arrows are measurement directions of the robot. (b): The colour indicates how many times the robot goes through one grid cell.

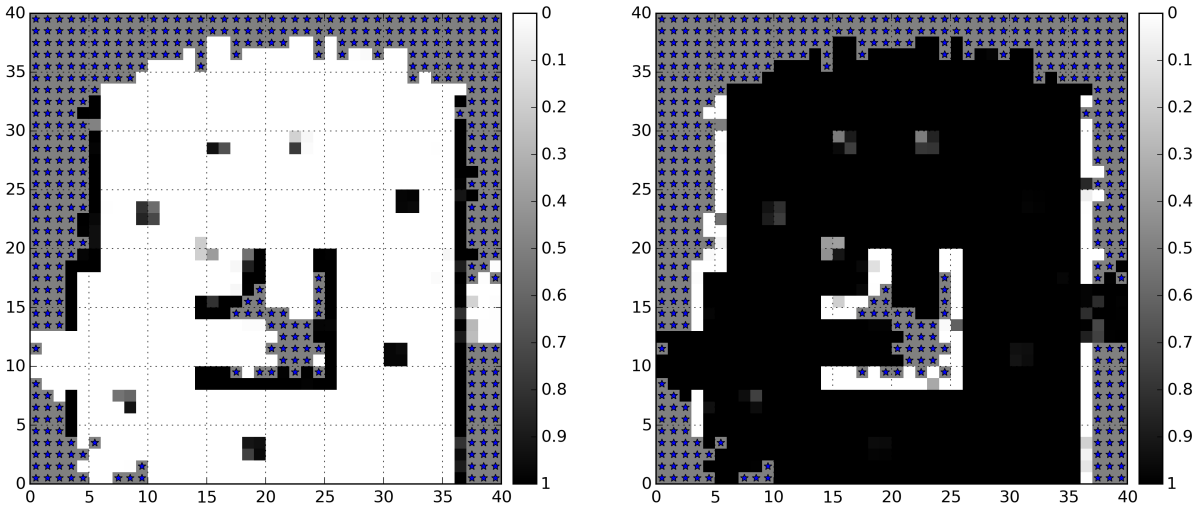
During one Δt, the map is assumed to be static. For computational convenience, the log observation defined in Section 4.2.2 is applied to build an occupancy grid map instead of the beam model of range sensors. The log observations O_i^t of grid cells observed occupied are 9, and the log observations O_i^t of grid cells observed free are -7. The initial log observation O_i^0 and the log observations O_i^t of unobserved grid cells are 0. One grid cell with a posterior occupancy probability more than 0.5 is assumed to be observed occupied once. One grid cell with a posterior occupancy probability less than 0.5 is assumed to be observed free once. After 20 loops, the total times that the grid cells are observed free or occupied are shown in Figure 5.4. In Figure 5.4(a), most of the free space is fully observed more than 20 times. It means some free grid cells are observed more than once during one loop. The walls have less chance to be observed free. As shown in Figure 5.4(b), the free space around objects and walls is observed occupied sometimes because of sensor noise. Most of the observed free space is never observed as being occupied.

Given the occupancy grid maps, the parameters of every grid cell are estimated individually. The initial values of parameters a_{11}^c, a_{22}^c, ρ_1^c are set to 0.5. The maximum number of iterations of the optimization process is set to 800. The estimates of parameters a_{11} and a_{22} for observed space are shown in Figure 5.5(a) and Figure 5.5(b), respectively. There is no estimate for unknown space covered by asterisks. In Figure 5.5(a), the estimate of a_{11} for free space is close to zero. It means free space will change to the free state from the occupied state quickly. The estimates for walls are high. It means the walls always stay occupied. The dynamic objects 1 and 2 change with high frequencies. They may change their states during one Δt. Their estimates of a_{11} are similar to dynamic objects 2 and 3. The dynamic objects 4 and 5 have higher estimates. The parameters of dynamic objects 6 and 7 are similar to the walls. Because of the sensor noise, some grid cells of the dynamic door are not estimated correctly, and the free space near the door has a higher estimate of a_{11} than most of the free space. As a whole, the dynamic object with a higher switching frequency has a lower estimate of a_{11}. In Figure 5.5(b), the estimates of free space and walls are opposite to their estimates in Figure 5.5(a). The free space has a high estimate of a_{22} and the walls have low estimates. For the dynamic objects, their estimates are similar to the corresponding estimates in Figure 5.5(a). The dynamic object with a higher switching frequency has a low estimate of a_{22}.



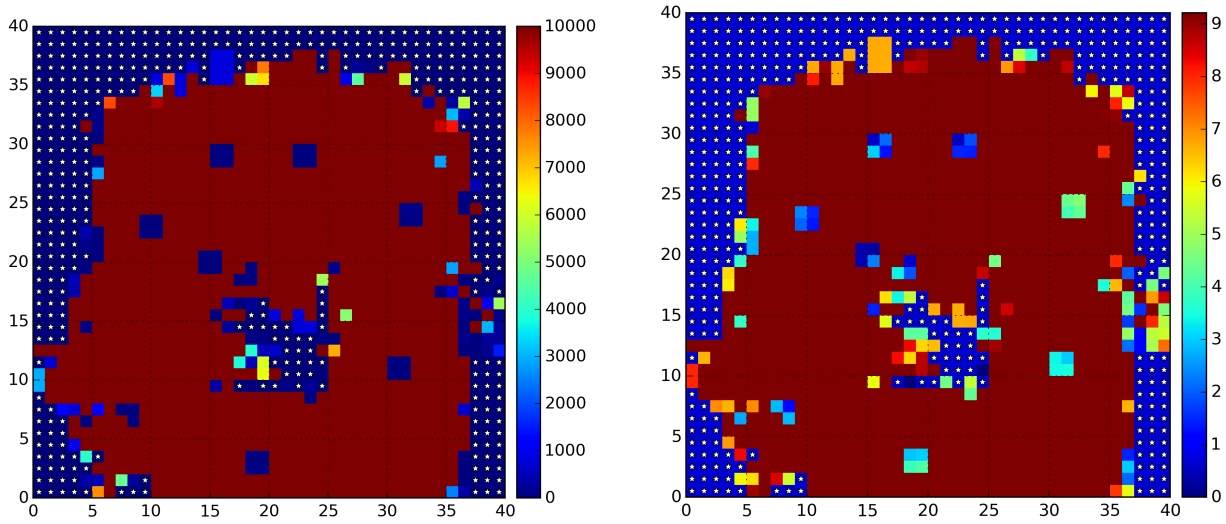
(a) Observed free (b) Observed occupied

Figure 5.4: Total observed times for the simulated dynamic environment. (a): The colour indicates the number of times one grid cell is observed free. (b): The colour indicates the number of times one grid cell is observed occupied.



(a) a_{11}: staying in occupied state (b) a_{22}: staying in free state

Figure 5.5: HMM parameter estimation without prior for the simulated dynamic environment. (a): The darkness indicates the probability of staying in occupied state. (b): The darkness indicates the probability of staying in free state.



(a) Overall expected duration

(b) Overall expected duration in log scale

Figure 5.6: Overall expected duration without prior for the simulated dynamic environment. (a): The colour indicates the overall expected duration. (b): The colour indicates the overall expected duration in log scale.

The overall expected duration is shown in Figure 5.6(a). The overall expected duration of the central free space is similar to the walls. The overall expected duration of the free space behind the door is short, and it is the same as the borders of observed free space. The difference between dynamic objects is not clear. In order to see the difference clearly, the overall expected duration in log scale is shown in Figure 5.6(b). The expected duration of the door is similar to the slow dynamic objects 7 and 8. The dynamic object with a low switching frequency has a long expected duration.

5.1.5 Discussion

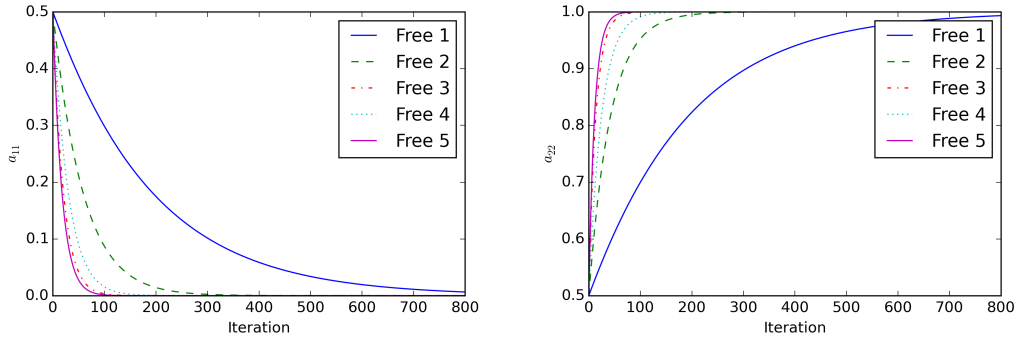
As shown in Figure 5.4, different grid cells may have different numbers of observations. Meanwhile, the convergence speeds of the parameters of different grid cells may also be different. In order to analyse the relationship between the number of observations and the parameter convergence speed, the following paragraphs discuss three kinds of grid cells: free, occupied, and dynamic.

For free grid cells, how the number of free observations influences the parameter convergence speed is introduced here. A free grid cell is the one with fewer occupied observations and more free observations. Table 5.1 shows the observation numbers of 5 free grid cells. All of them have no occupied observation and have different numbers of free observations. The optimization processes of their parameters are shown in Figure 5.7. With more free observations, the optimization process converges faster. The derivatives of Q function with respect to a_{11} and a_{22} are shown in Figure 5.8. Even though there is no occupied observation, the derivatives of Q function with respect to a_{11} converge to zeros. When the parameters a_{22} converge to 1, the derivatives of Q function with respect to a_{22} converge to nonzero constants. The best estimates of a_{22} for these free grid cells are 1. When a_{22} reaches 0.9, the corresponding derivative of Q function with respect to a_{11} is close to 0.

How the number of occupied observations influences the parameter convergence speed for occupied grid cells is introduced in this paragraph. As one occupied grid cell, it should have fewer free observations and

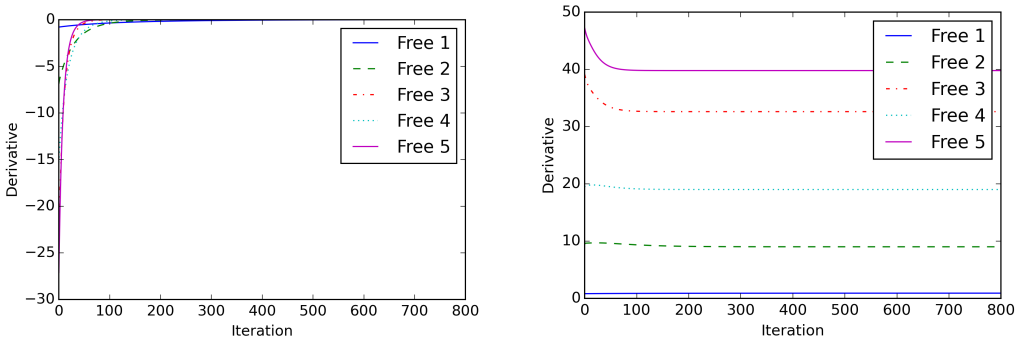
Table 5.1: Observation numbers of the selected free grid cells

Free grid cell	1	2	3	4	5
Free observation number	2	11	22	35	44
Occupied observation number	0	0	0	0	0



(a) a_{11}: staying in occupied state (b) a_{22}: staying in free state

Figure 5.7: Optimization process of the parameters of the selected free grid cells



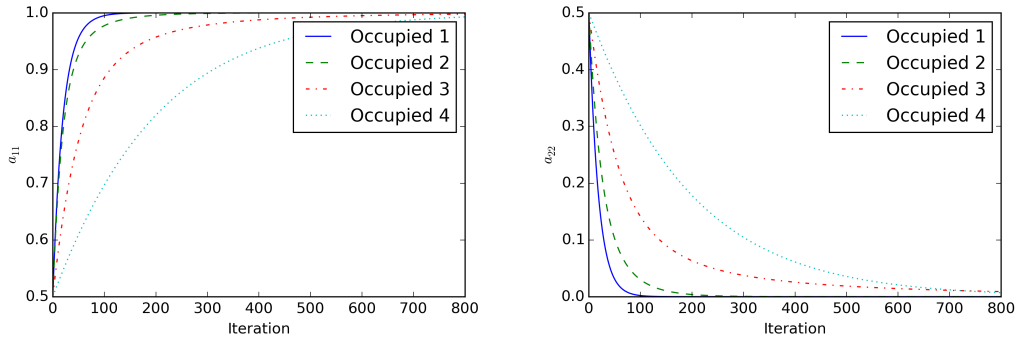
(a) Derivative of Q with respect to a_{11} (b) Derivative of Q with respect to a_{22}

Figure 5.8: Optimization process of the parameter derivatives of the selected free grid cells

more occupied observations. Table 5.2 shows the observation numbers of 4 occupied grid cells and most of their observations are occupied. Because of noise, it is not easy to choose occupied grid cells without free observation. As a result, the first three occupied grid cells have 1 or 2 free observations. The corresponding optimization processes are shown in Figure 5.9. With more occupied observations, the optimization process also converges faster. Even though there is no free observation for occupied grid cell 4, the parameter a_{22} can also be optimized by occupied observations. As shown in Figure 5.10, the derivatives of Q function with respect to a_{11} and a_{22} converge to nonzero constants and zeros, respectively. Similarly, the best estimates of a_{11} for these occupied points are 1. When a_{11} reaches 0.9, the corresponding derivative of Q function with respect to a_{22} is close to 0.

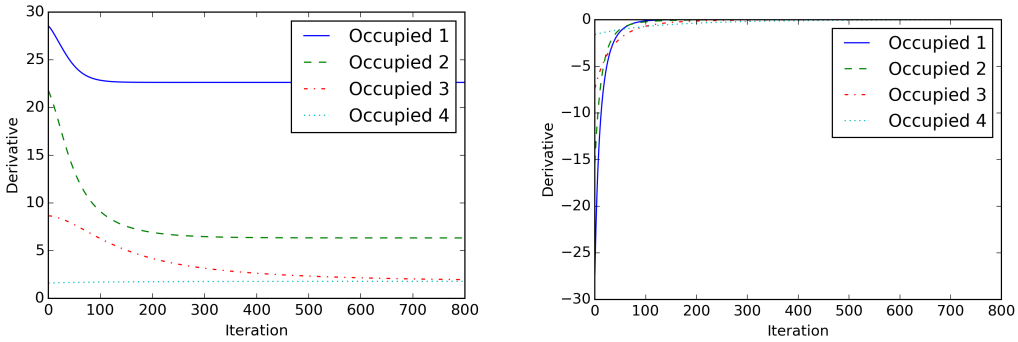
Table 5.2: Observation numbers of the selected occupied grid cells

Occupied grid cell	1	2	3	4
Free observation number	1	2	1	0
Occupied observation number	34	26	12	3



(a) a_{11}: staying in occupied state (b) a_{22}: staying in free state

Figure 5.9: Optimization process of the parameters of the selected occupied grid cells



(a) Derivative of Q with respect to a_{11} (b) Derivative of Q with respect to a_{22}

Figure 5.10: Optimization process of the parameter derivatives of the selected occupied grid cells

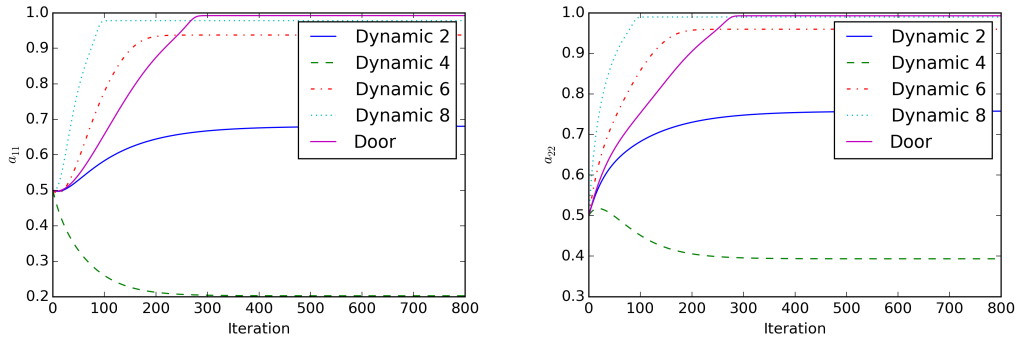
As to dynamic grid cells, the dynamic objects 2, 4, 6, 8 and the door are taken as examples, and their observation numbers are shown in Table 5.3. They change with different frequencies, and their numbers of free observations are similar to the corresponding numbers of occupied observations. For one dynamic object, the estimations of a_{11} and a_{22} converge at similar speeds. In Figure 5.12, all the derivatives converge to zeros.

Table 5.3: Observation numbers of the selected dynamic objects

Dynamic object	2	4	6	8	door
Free observation number	16	21	16	24	12
Occupied observation number	17	17	12	13	8

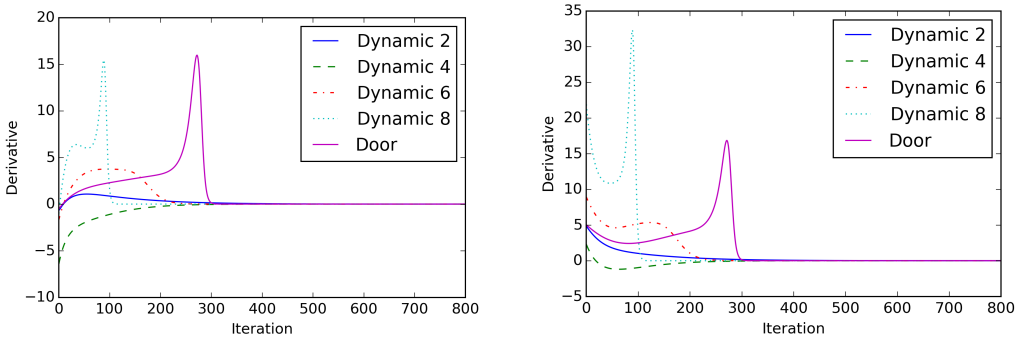
5.2 Prediction based on Markov random fields

In the previous section, Markov chains are applied to model dynamic environments, and there are only two parameters in a transition matrix. In this section, grid maps are applied to represent dynamic environments, and every grid cell is described by two parameters of a Markov chain. Based on MRF theory, a new method is proposed to estimate the parameters of observed and unobserved space.



(a) a_{11}: staying in occupied state (b) a_{22}: staying in free state

Figure 5.11: Optimization process of the parameters of the selected dynamic objects



(a) Derivative of Q with respect to a_{11} (b) Derivative of Q with respect to a_{22}

Figure 5.12: Optimization process of the parameter derivatives of the selected dynamic objects

5.2.1 The Markov random field model

For one grid cell, the probabilities of staying in occupied and free states are denoted by a_{11}^c and a_{22}^c, respectively. The set of a_{11}^c and a_{22}^c for all the grid cells is denoted by $\boldsymbol{A} = (\boldsymbol{a}_1, \boldsymbol{a}_2)$, where $\boldsymbol{a}_1 = [\cdots, a_{11}^c, \cdots]^\top$ and $\boldsymbol{a}_2 = [\cdots, a_{22}^c, \cdots]^\top$. Assuming $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are independent, the prior distribution can be factorized as

$$p(\boldsymbol{A}) = p(\boldsymbol{a}_1)p(\boldsymbol{a}_2), \tag{5.24}$$

where $p(\boldsymbol{a}_1)$ and $p(\boldsymbol{a}_2)$ are assumed to have the same distribution. The prior distribution $p(\boldsymbol{a}_1)$ is taken as an example to derive the prior distribution which also takes the advantage of log odds form. The log odds form of a_{11}^c is defined by

$$l_{a_{11}}^c = \log \frac{a_{11}^c}{1 - a_{11}^c}. \tag{5.25}$$

The vector of all the a_{11}^c in log odds form is denoted by $\boldsymbol{l}_{a_1} = [\cdots, l_{a_{11}}^c, \cdots]^\top$ and can be expressed by

$$\boldsymbol{l}_{a_1} = \log \frac{\boldsymbol{a}_1}{\boldsymbol{1} - \boldsymbol{a}_1}, \tag{5.26}$$

where $\boldsymbol{1}$ is a column of ones and the division is elementwise. The vector $\boldsymbol{l}_{a_1}$ is regarded as an MRF where only the pair-variable cliques in the second-order neighbourhood are considered. Similarly to equation

(4.24), the prior distribution $p(\boldsymbol{a_1})$ is

$$p(\boldsymbol{a}_1) = \frac{1}{Z}\exp\left(-\frac{2}{\mathcal{T}}\boldsymbol{l}_{a_1}^{\top}\boldsymbol{\mathcal{A}}\boldsymbol{l}_{a_1}\right) \tag{5.27}$$

$$= \frac{1}{Z}\exp\left(-\frac{1}{\mathcal{T}}U(\boldsymbol{a}_1)\right), \tag{5.28}$$

where

$$U(\boldsymbol{a}_1) = 2(\log\frac{\boldsymbol{a}_1}{\boldsymbol{1}-\boldsymbol{a}_1})^{\top}\boldsymbol{\mathcal{A}}\log\frac{\boldsymbol{a}_1}{\boldsymbol{1}-\boldsymbol{a}_1}, \tag{5.29}$$

$$Z = \sum_{\boldsymbol{a}_1}\exp\left(-\frac{1}{\mathcal{T}}U(\boldsymbol{a}_1)\right). \tag{5.30}$$

The temperature $\mathcal{T}$ has a role of weight between the prior distribution and the likelihood, and Z is a normalizer. The matrix $\boldsymbol{\mathcal{A}}$ depends on the cliques in MRFs. Similarly, the log odds form of a_{22}^c is defined by

$$l_{a_{22}}^c = \log\frac{a_{22}^c}{1-a_{22}^c}. \tag{5.31}$$

The vector of all the $l_{a_{22}}^c$ is denoted by $\boldsymbol{l}_{a_2} = [\cdots, l_{a_{22}}^c, \cdots]^{\top}$ and regarded as the same MRF. By analogy with equation (5.28), the prior distribution $p(\boldsymbol{a}_2)$ is

$$p(\boldsymbol{a}_2) = \frac{1}{Z}\exp\left(-\frac{1}{\mathcal{T}}U(\boldsymbol{a}_2)\right), \tag{5.32}$$

where

$$U(\boldsymbol{a}_2) = 2(\log\frac{\boldsymbol{a}_2}{\boldsymbol{1}-\boldsymbol{a}_2})^{\top}\boldsymbol{\mathcal{A}}\log\frac{\boldsymbol{a}_2}{\boldsymbol{1}-\boldsymbol{a}_2}. \tag{5.33}$$

Since $p(\boldsymbol{a}_1)$ and $p(\boldsymbol{a}_2)$ have the same distribution, the normalizers Z are the same.

Assuming the coordinate set of observed grid cells is denoted by $\mathcal{I}$ and the states of different grid cells are independent, the map in time instant t that consists of the states of observed grid cells is denoted by $\mathcal{M}^t = \{m_c^0, \cdots, m_c^t, \cdots\}(c \in \mathcal{I})$ and its distribution is described as

$$p(\mathcal{M}^t) = \prod_{c\in\mathcal{I}} p(m_c^t). \tag{5.34}$$

A map sequence consists of different maps $\mathcal{M}^t$ in time and is denoted by $\mathcal{M} = \{\mathcal{M}^0, \cdots, \mathcal{M}^t, \cdots\}$. As shown in Figure 5.13, the map sequence depends on space and time. Meanwhile, the map sequence also consists of different state sequences $\mathcal{M}_c(c \in \mathcal{I})$ in space and can also be expressed as $\mathcal{M} = \{\cdots, \mathcal{M}_c, \cdots\}(c \in \mathcal{I})$. Because of the state independence between different grid cells, the distribution of the map sequence $\mathcal{M}$ is computed by

$$p(\mathcal{M}) = \prod_{c\in\mathcal{I}} p(\mathcal{M}_c). \tag{5.35}$$

As mentioned in the previous section, one robot may have several measurements z_i^t in one time instant t. These measurements are denoted by a sequence $y_t = (z_1^t, z_2^t, \cdots)$, and all the measurements are denoted by an observation sequence $O = (y_0, \cdots, y_t, \cdots)$. The likelihood of $\boldsymbol{A}$ given the observation sequence O

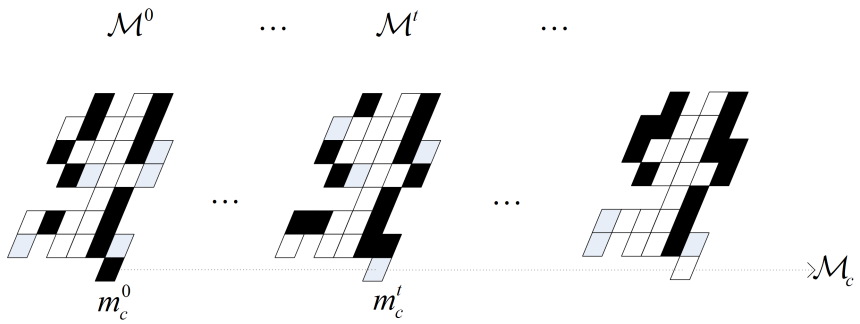


Figure 5.13: An example of map sequences. One map sequence consists of different maps in time or different state sequences in space.

and the underlying map sequence $\mathcal{M}$ is

$$p(O, \mathcal{M} \mid \boldsymbol{A}) = p(O \mid \mathcal{M}, \boldsymbol{A}) p(\mathcal{M} \mid \boldsymbol{A}) \tag{5.36}$$
$$= p(O \mid \mathcal{M}) p(\mathcal{M} \mid \boldsymbol{A}), \tag{5.37}$$

Given the map sequence $\mathcal{M}$, the observation sequence O is conditionally independent of the parameters $\boldsymbol{A}$. Assuming observations are independent, the probability $p(O \mid \mathcal{M})$ can be obtained by

$$p(O \mid \mathcal{M}) = \prod_t p(y_t \mid \mathcal{M}^t), \tag{5.38}$$

where $p(y_t \mid \mathcal{M}^t)$ is the probability of the measurement sequence y_t conditional on the map $\mathcal{M}^t$ in time instant t. The measurement sequence y_t only depends on the map $\mathcal{M}^t$ in the same time instant t. The probability $p(y_t \mid \mathcal{M}^t)$ can be derived from the measurement model $p(z_t^i \mid \mathcal{M}^t)$,

$$p(y_t \mid \mathcal{M}^t) = \prod_i p(z_t^i \mid \mathcal{M}^t). \tag{5.39}$$

Because of the state independence between different grid cells, the state sequence $\mathcal{M}_c$ only depends on the corresponding parameter A_c at the same coordinate c, and the probability $p(\mathcal{M} \mid \boldsymbol{A})$ can be factorized as

$$p(\mathcal{M} \mid \boldsymbol{A}) = \prod_{c \in \mathcal{I}} p(\mathcal{M}_c \mid A_c). \tag{5.40}$$

The likelihood function is

$$p(O \mid \boldsymbol{A}) = \sum_{\mathcal{M}} p(O, \mathcal{M} \mid \boldsymbol{A}), \tag{5.41}$$

where every $\mathcal{M}$ only contains the observed grid cells. The unobserved space is not considered in the likelihood function. The observation sequence is also conditional on the initial map distribution $p(\mathcal{M}^0)$ which requires the initial state probabilities $\rho_i^c = p(m_c^0 = s_i)$ of observed grid cells as in equation (5.34). For convenience, the initial probabilities are not written together with $\boldsymbol{A}$.

Finally, based on the Bayes rule, the MRF model is

$$p(\boldsymbol{A} \mid O) = \frac{p(O \mid \boldsymbol{A}) p(\boldsymbol{A})}{p(O)}, \tag{5.42}$$

where $p(O)$ is a constant.

5.2.2 Parameter estimation

The parameters can be obtained by maximizing $p(\boldsymbol{A} \mid O)$, or equivalently maximizing

$$p(O, \boldsymbol{A}) = p(O \mid \boldsymbol{A})p(\boldsymbol{A}). \tag{5.43}$$

This problem is similar to the HMM case without prior, and its direct maximization is not feasible because equation (5.41) requires summing over all possible maps. The EM algorithm can applied to recursively maximize the Q function (see Appendix B)

$$\begin{aligned} Q(\boldsymbol{A}, \boldsymbol{A}^{(k)}) &= E_{\mathcal{M}|O,\boldsymbol{A}^{(k)}} \log p(\mathcal{M}, O \mid \boldsymbol{A}) + \log p(\boldsymbol{A}) && (5.44) \\ &= E_{\mathcal{M}|O,\boldsymbol{A}^{(k)}} \log p(O \mid \mathcal{M}, \boldsymbol{A}) + E_{\mathcal{M}|O,\boldsymbol{A}^{(k)}} \log p(\mathcal{M} \mid \boldsymbol{A}) \\ &\quad - 2\log Z - \frac{1}{\mathcal{T}}U(\boldsymbol{a}_1) - \frac{1}{\mathcal{T}}U(\boldsymbol{a}_2), && (5.45) \end{aligned}$$

where $\boldsymbol{A}^{(k)}$ represents the parameters obtained in iteration k. Since the observation sequence O is conditionally independent of the parameters given the map sequence $\mathcal{M}$, the probability $p(O \mid \mathcal{M}, \boldsymbol{A})$ can be rewritten as $p(O \mid \mathcal{M})$ and is a constant. The normalizer Z is also a constant. As a result, the parameters can be obtained by maximizing the non-constant terms. Discarding the constant terms gives

$$\begin{aligned} &E_{\mathcal{M}|O,\boldsymbol{A}^{(k)}} \log p(\mathcal{M} \mid \boldsymbol{A}) - \frac{1}{\mathcal{T}}U(\boldsymbol{a}_1) - \frac{1}{\mathcal{T}}U(\boldsymbol{a}_2) \\ &= \sum_{\mathcal{M}} p(\mathcal{M} \mid O, \boldsymbol{A}^{(k)}) \sum_{c\in\mathcal{I}} \log p(\mathcal{M}_c \mid A_c) - \frac{1}{\mathcal{T}}U(\boldsymbol{a}_1) - \frac{1}{\mathcal{T}}U(\boldsymbol{a}_2) \\ &= \sum_{c\in\mathcal{I}} \sum_{\mathcal{M}} p(\mathcal{M} \mid O, \boldsymbol{A}^{(k)}) \log p(\mathcal{M}_c \mid A_c) - \frac{1}{\mathcal{T}}U(\boldsymbol{a}_1) - \frac{1}{\mathcal{T}}U(\boldsymbol{a}_2) \\ &= \sum_{c\in\mathcal{I}} \sum_{\mathcal{M}_c} p(\mathcal{M}_c \mid O, A_c^{(k)}) \log p(\mathcal{M}_c \mid A_c) - \frac{1}{\mathcal{T}}U(\boldsymbol{a}_1) - \frac{1}{\mathcal{T}}U(\boldsymbol{a}_2). \end{aligned} \tag{5.46}$$

As mentioned in the previous section, the initial state probabilities ρ_i^c are not written together with A_c. The probability $p(\mathcal{M}_c \mid O, A_c^{(k)})$ is the same as $p(\mathcal{M}_c \mid O, \theta_c^{(k)})$ in equation (5.10), where θ_c includes A_c and the initial state probabilities ρ_i^c. Therefore equation (5.46) can be rewritten as

$$\begin{aligned} &E_{\mathcal{M}|O,\boldsymbol{A}^{(k)}} \log p(\mathcal{M} \mid \boldsymbol{A}) - \frac{1}{\mathcal{T}}U(\boldsymbol{a}_1) - \frac{1}{\mathcal{T}}U(\boldsymbol{a}_2) \\ &= \sum_{c\in\mathcal{I}} f(\rho_1^c) + \sum_{c\in\mathcal{I}} f(a_{11}^c) + \sum_{c\in\mathcal{I}} f(a_{22}^c) - \frac{1}{\mathcal{T}}U(\boldsymbol{a}_1) - \frac{1}{\mathcal{T}}U(\boldsymbol{a}_2) \\ &= f(\boldsymbol{\rho}_1) + f(\boldsymbol{a}_1) + f(\boldsymbol{a}_2) \end{aligned} \tag{5.47}$$

where $\boldsymbol{\rho}_1$ is defined by $\boldsymbol{\rho}_1 = [\cdots, \rho_1^c, \cdots]^\top$ which is the vector of the initial occupancy probabilities ρ_1^c of observed grid cells. The function $f(\boldsymbol{\rho}_1)$, $f(\boldsymbol{a}_1)$ and $f(\boldsymbol{a}_2)$ are the vector versions of the ones defined in equations (5.11), (5.12) and (5.13), defined by

$$\begin{aligned} f(\boldsymbol{\rho}_1) &= \sum_{c\in\mathcal{I}} f(\rho_1^c) \\ &= \boldsymbol{\gamma}_1 \log \boldsymbol{\rho}_1 + \boldsymbol{\gamma}_2 \log (\mathbf{1} - \boldsymbol{\rho}_1), \end{aligned} \tag{5.48}$$

$$f(\boldsymbol{a}_1) = \sum_c f(a_{11}^c) - \frac{1}{\mathcal{T}} U(\boldsymbol{a}_1)$$
$$= \boldsymbol{\xi}_{11} \log \boldsymbol{a}_1 + \boldsymbol{\xi}_{12} \log (\mathbf{1} - \boldsymbol{a}_1) - \frac{2}{\mathcal{T}} (\log \frac{\boldsymbol{a}_1}{\mathbf{1} - \boldsymbol{a}_1})^\top \boldsymbol{\mathcal{A}} \log \frac{\boldsymbol{a}_1}{\mathbf{1} - \boldsymbol{a}_1}, \tag{5.49}$$

$$f(\boldsymbol{a}_2) = \sum_c f(a_{22}^c) - \frac{1}{\mathcal{T}} U(\boldsymbol{a}_2)$$
$$= \boldsymbol{\xi}_{22} \log \boldsymbol{a}_2 + \boldsymbol{\xi}_{21} \log (\mathbf{1} - \boldsymbol{a}_2) - \frac{2}{\mathcal{T}} (\log \frac{\boldsymbol{a}_2}{\mathbf{1} - \boldsymbol{a}_2})^\top \boldsymbol{\mathcal{A}} \log \frac{\boldsymbol{a}_2}{\mathbf{1} - \boldsymbol{a}_2}. \tag{5.50}$$

The vector $\boldsymbol{\gamma}_i = [\cdots, \gamma_i^c(0), \cdots]$, where $\gamma_i^c(0) = p(m_c^0 = s_i \mid O, \theta_c^{(k)})$ is the initial probability of being in state s_i given the observation sequence O. The vector $\boldsymbol{\xi}_{ij} = [\cdots, \sum_t \xi_{ij}^c(t), \cdots]$, where $\xi_{ij}^c(t) = p(m_c^t = s_i, m_c^{t+1} = s_j \mid O, \theta_c^{(k)})$ which is the probability of being in state s_i in time instant t and s_j in time instant $t+1$ given the observation sequence O. The sums in three functions are rewritten as vector products. Both $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ consist of the parameters in observed space and unobserved space. In equations (5.49) and (5.50), the vector $\boldsymbol{\xi}_{ij}$ is obtained from the forward-backward algorithm and has the same dimensions of $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$. For unobserved grid cells, the corresponding elements in $\boldsymbol{\xi}_{ij}$ are set to 0. The first two terms in equations (5.49) and (5.50) come from the likelihood function and the last one comes from the prior. There is no prior knowledge for $\boldsymbol{\rho}_1$. The three functions contains different parameters $\boldsymbol{\rho}_1$, $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$. The parameters can be obtained by maximizing the three functions individually. The derivatives of $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(k)})$ with respect to the parameters $\boldsymbol{\rho}_1$, $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are given by

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\rho}_1} f(\boldsymbol{\rho}_1) = \boldsymbol{\gamma}_1^\top \oslash \boldsymbol{\rho}_1 - \boldsymbol{\gamma}_2^\top \oslash (\mathbf{1} - \boldsymbol{\rho}_1), \tag{5.51}$$

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{a}_1} f(\boldsymbol{a}_1) = \boldsymbol{\xi}_{11}^\top \oslash \boldsymbol{a}_1 - \boldsymbol{\xi}_{12}^\top \oslash (\mathbf{1} - \boldsymbol{a}_1) - \frac{4}{\mathcal{T}} \boldsymbol{\mathcal{A}} (\log \frac{\boldsymbol{a}_1}{\mathbf{1} - \boldsymbol{a}_1}) \odot \frac{\mathbf{1}}{\boldsymbol{a}_1 \odot (\mathbf{1} - \boldsymbol{a}_1)}, \tag{5.52}$$

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{a}_2} f(\boldsymbol{a}_2) = \boldsymbol{\xi}_{22}^\top \oslash \boldsymbol{a}_2 - \boldsymbol{\xi}_{21}^\top \oslash (\mathbf{1} - \boldsymbol{a}_2) - \frac{4}{\mathcal{T}} \boldsymbol{\mathcal{A}} (\log \frac{\boldsymbol{a}_2}{\mathbf{1} - \boldsymbol{a}_2}) \odot \frac{\mathbf{1}}{\boldsymbol{a}_2 \odot (\mathbf{1} - \boldsymbol{a}_2)}, \tag{5.53}$$

where $\oslash$ is the elementwise division and $\odot$ is the elementwise product (Hadamard product). Without prior knowledge in equation (5.51), the function $f(\boldsymbol{\rho}_1)$ can be maximized directly. Since $\boldsymbol{\gamma}_1^\top + \boldsymbol{\gamma}_2^\top = \mathbf{1}$, the estimation of $\boldsymbol{\rho_1}$ is

$$\boldsymbol{\rho_1} = \boldsymbol{\gamma}_1^\top. \tag{5.54}$$

With the prior distributions $p(\boldsymbol{a}_1)$ and $p(\boldsymbol{a}_2)$, it is not easy to maximize the two functions $f(\boldsymbol{a}_1)$ and $f(\boldsymbol{a}_2)$, or equivalently to minimize $-f(\boldsymbol{a}_1)$ and $-f(\boldsymbol{a}_2)$. Since this estimation is only one step in the whole optimization process, the simple line search method (LSM) described as Algorithm 7 is used to estimate $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ in range (0,1). The LSM is a gradient-based method where there are three inputs: the objective function, the initial value vector of parameters and the maximum optimizing iteration $step$. This method searches for the optimum of the objective function from the initial values of parameters iteratively. Because only the first iterations are numerically relevant, the LSM can be stopped before convergence in order to achieve computationally efficiency. As discussed in Section 5.1.5, the derivative of Q with respect to a_{11} for occupied space and the derivative of Q with respect to a_{22} for free space never converge to zeros. The LSM will always search for the estimates for them even though their estimates are close to 1. In order to give more chance to other parameters, the optimization process stops searching for them when their estimates reach 0.995. The grid cells with more observations will converge faster. For the border of observed space

and unobserved space, there are fewer or no observations, and it will take a long time to converge.

5.2.3 Computational complexity

This MRF-based prediction in dynamic environments is described as the Algorithm 5 below.

Algorithm 5 Prediction based on Markov random fields

Input: Measurements z_i^t
Output: $\boldsymbol{a}_1^{(k)}$ and $\boldsymbol{a}_2^{(k)}$
1: Initialize $step$, $\boldsymbol{\rho}_1$, $\boldsymbol{a}_1$, $\boldsymbol{a}_2$ and $k = 0$
2: Build occupancy grid maps with temporary state probabilities 0.5 and obtain $p(m_c^t \mid y_t)$
3: **while** $k <$ maximum iteration **do**
4: Calculate $\boldsymbol{\gamma}_1$ and $\boldsymbol{\xi}_{ij}$ using equations (5.22) and (5.23)
5: $\boldsymbol{\rho}_1^{(k+1)} = \boldsymbol{\gamma}_1^{\top}$
6: $\boldsymbol{a}_1^{(k+1)}$ =LSM$(-f(\boldsymbol{a}_1), \boldsymbol{a}_1^{(k)}, step)$
7: $\boldsymbol{a}_2^{(k+1)}$ =LSM$(-f(\boldsymbol{a}_2), \boldsymbol{a}_2^{(k)}, step)$
8: $k = k + 1$

The computational complexity is obtained as follows.

Step 2 The problems with different numbers of parameters have different maximum iterations.

Step 3 As described in Algorithm 4, the computational complexity of this step is $\mathcal{O}(\zeta n_o)$, where ζ is the length of the observation sequence O and n_o is the number of observed grid cells.

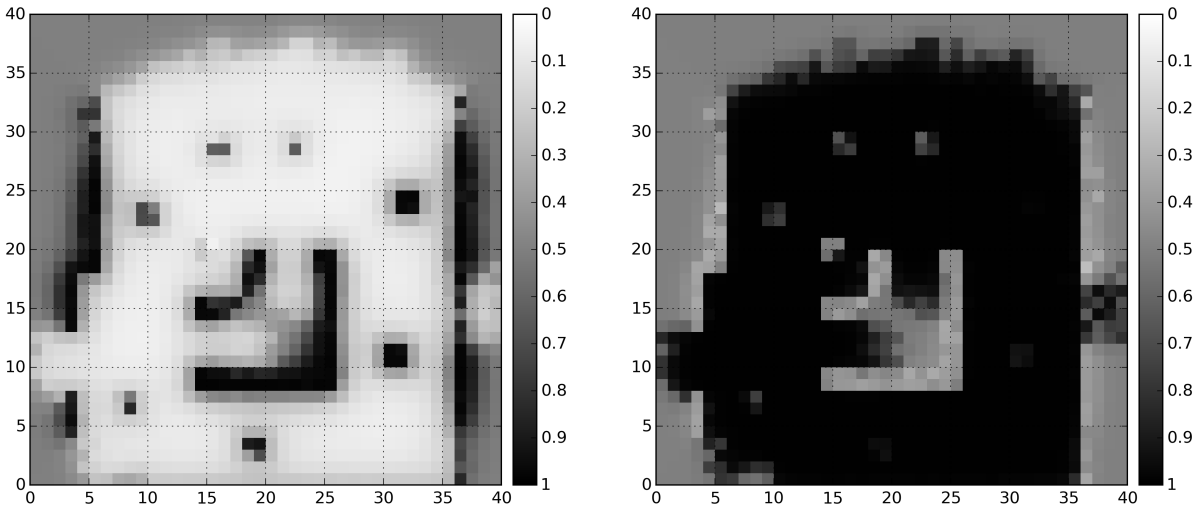
Step 4 For every observed grid cell, the forward-backward procedure has a complexity of $\mathcal{O}(\zeta)$. The complexity of this step is $\mathcal{O}(\zeta n_o)$.

Steps 6 and 7 Two functions and their derivatives need to be computed by equations (5.49), (5.50), (5.52) and (5.53). The computational complexities of these two steps are the same as $\mathcal{O}(n^2)$, where n is the number of all the grid cells.

5.2.4 Simulation

The observation data simulated in Section 5.1 is used to test this method. The initial parameters $\boldsymbol{a}_1, \boldsymbol{a}_2$, and $\boldsymbol{\rho}_1$ are 0.5, and the temperature $\mathcal{T} = 125$. The maximum number of iterations of the optimization process is set to 300 and the parameter estimation is shown in Figure 5.14. Figure 5.14(a) is similar to Figure 5.3(a) and smoother. In Figure 5.14(b), the estimate of free space is similar to 5.3(b). For the walls, there are none or fewer free observations. It is not easy to estimate the probability of staying in free state. In the prior knowledge, the walls depend on their neighbours which are free space and unobserved space. The parameter a_{22} of the free space is more than 0.5 and the parameter a_{22} of unobserved space is close to 0.5. As a result, the parameters of the walls, which should be close to 0, are a little high. The parameters converge very slowly for unobserved space. The parameters of most of the unobserved space are the same as the initial values 0.5, and only the parameters of the unobserved space near the observed space is optimized.

The overall expected duration is shown in Figure 5.15(a), which is also smoother. The overall expected duration of free space is very long, and the overall expected durations of dynamic objects are very short.

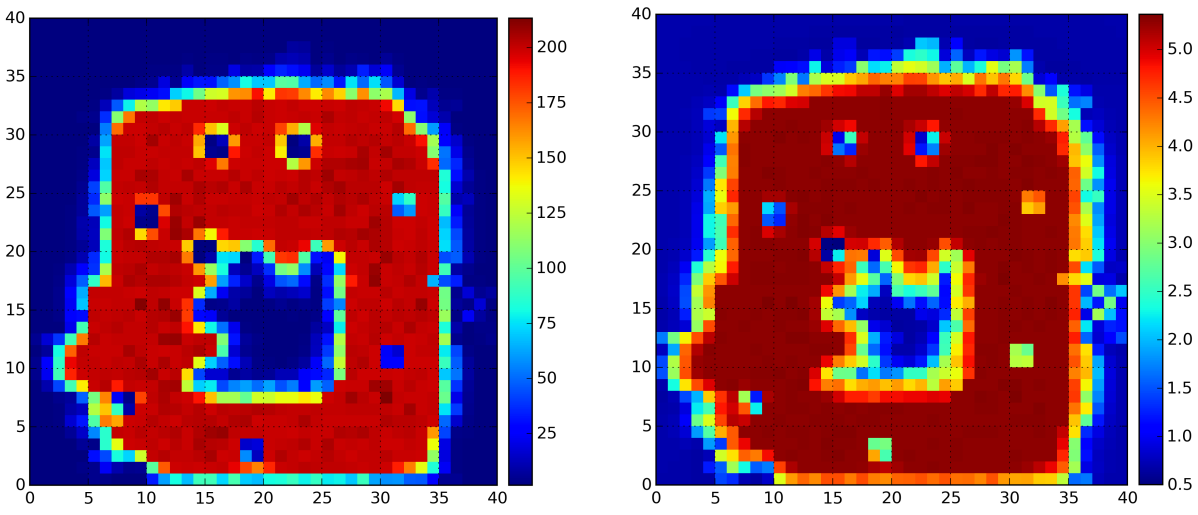


(a) a_{11}: staying in occupied state

(b) a_{22}: staying in free state

Figure 5.14: HMM parameter estimation of the MRF-based prediction for the simulated dynamic environment

The overall expected duration in log scale is shown in Figure 5.15(b). The difference between dynamic objects is shown more clearly, and slower dynamic objects have longer expected durations. The parameters of most of the unobserved space are not optimized, and the corresponding expected duration is very short.



(a) Overall expected duration

(b) Overall expected duration in log scale

Figure 5.15: Overall expected duration of the MRF-based prediction for the simulated dynamic environment

5.3 Prediction based on Gaussian random fields

In the previous section, a method based on MRFs is proposed. In this section, the MRF-based prior knowledge is replaced by GRFs which works in continuous space. Like before, each point is associated with two HMM parameters, a_{11} and a_{22}, which are the probabilities of staying in occupied and free states. This means there will be two random fields, one for each parameter.

In order to obtain the observation sequence, the observed space is also divided into grid cells. When any point in one grid cell is observed, the central point is assumed to be observed once. Given an observation sequence, the parameters of any point in continuous space can be estimated. Training data consists of the central points of the grid cells. The coordinate set of training points and test points are denoted by $\mathcal{I}$ and $\mathcal{I}_*$, respectively. The parameters of training points are denoted by $\boldsymbol{A} = (\boldsymbol{a}_{11}, \boldsymbol{a}_{22})$, where $\boldsymbol{a}_{11} = [\cdots, a_{11}^c, \cdots]^\top (c \in \mathcal{I})$ and $\boldsymbol{a}_{22} = [\cdots, a_{22}^c, \cdots]^\top (c \in \mathcal{I})$. The parameters of test points are denoted by $\boldsymbol{A}_* = (\boldsymbol{a}_{11}^*, \boldsymbol{a}_{22}^*)$, where $\boldsymbol{a}_{11}^* = [\cdots, a_{11}^c, \cdots]^\top (c \in \mathcal{I}_*)$ and $\boldsymbol{a}_{22}^* = [\cdots, a_{22}^c, \cdots]^\top (c \in \mathcal{I}_*)$. In probabilistic form, the parameter estimation of all the selected points including training and test points is

$$p(\boldsymbol{A}, \boldsymbol{A}_* \mid O). \tag{5.55}$$

The problem can be factorized as

$$p(\boldsymbol{A}, \boldsymbol{A}_* \mid O) = p(\boldsymbol{A}_* \mid \boldsymbol{A}) p(\boldsymbol{A} \mid O), \tag{5.56}$$

where $p(\boldsymbol{A} \mid O)$ is an HMM parameter estimation problem for training points and $p(\boldsymbol{A}_* \mid \boldsymbol{A})$ is an HMM parameter prediction problem for test points. The parameter estimation for training and test points are done individually.

5.3.1 Hidden Markov model parameter estimation

Assuming two parameters a_{11}^c and a_{22}^c are independent, the prior distribution can be factorized as

$$p(\boldsymbol{A}) = p(\boldsymbol{a}_{11}) p(\boldsymbol{a}_{22}). \tag{5.57}$$

The distributions $p(\boldsymbol{a}_{11})$ and $p(\boldsymbol{a}_{22})$ are assumed to be the same and $p(\boldsymbol{a}_{11})$ is taken as an example to derive the prior distribution. The log odds form $l_{a_{11}}^c$ of a_{11}^c is also applied to formulate the prior distribution, which is defined by equation (5.25). The vector of all the $l_{a_{11}}^c$ of training points is denoted by $\boldsymbol{l}_{a_{11}} = [\cdots, l_{a_{11}}^c, \cdots]^\top (c \in \mathcal{I})$ and assumed to be Gaussian distributed as

$$\boldsymbol{l}_{a_{11}} \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{K}_{\mathcal{I}\mathcal{I}}), \tag{5.58}$$

where $\boldsymbol{\mu}_1$ is the mean vector and $\boldsymbol{K}_{\mathcal{I}\mathcal{I}}$ is the covariance matrix. The covariance function is Ornstein–Uhlenbeck kernel function as in equation (4.57). The prior distribution $p(\boldsymbol{a}_{11})$ is

$$p(\boldsymbol{a}_{11}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{K}_{\mathcal{I}\mathcal{I}}|}} \exp\left(-\frac{1}{2} U(\boldsymbol{a}_{11})\right), \tag{5.59}$$

where n is the number of training points and

$$U(\boldsymbol{a}_{11}) = (\log \frac{\boldsymbol{a}_{11}}{\boldsymbol{1} - \boldsymbol{a}_{11}} - \boldsymbol{\mu}_1)^\top \boldsymbol{K}_{\mathcal{I}\mathcal{I}}^{-1} (\log \frac{\boldsymbol{a}_{11}}{\boldsymbol{1} - \boldsymbol{a}_{11}} - \boldsymbol{\mu}_1). \tag{5.60}$$

The log odds form $l^c_{a_{22}}$ of a^c_{22} is defined by equation (5.31) and the vector of all the $l^c_{a_{22}}$ of training points is denoted by $\boldsymbol{l}_{a_{22}} = [\cdots, l^c_{a_{22}}, \cdots]^\top (c \in \mathcal{I})$, which is also assumed to be Gaussian distributed. The prior distribution $p(\boldsymbol{a}_{22})$ is given in the same way by

$$p(\boldsymbol{a}_{22}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{K}_{\mathcal{I}\mathcal{I}}|}} \exp\left(-\frac{1}{2} U(\boldsymbol{a}_{22})\right), \tag{5.61}$$

where

$$U(\boldsymbol{a}_{22}) = (\log \frac{\boldsymbol{a}_{22}}{\boldsymbol{1} - \boldsymbol{a}_{22}} - \boldsymbol{\mu}_2)^\top \boldsymbol{K}^{-1}_{\mathcal{I}\mathcal{I}} (\log \frac{\boldsymbol{a}_{22}}{\boldsymbol{1} - \boldsymbol{a}_{22}} - \boldsymbol{\mu}_2), \tag{5.62}$$

and $\boldsymbol{\mu}_2$ is the mean vector of $\boldsymbol{l}_{a_{22}}$.

In one time instant t, there may be several measurements z^t_i which are denoted by a sequence $y_t = (z^t_1, z^t_2, \cdots)$. Assuming all the measurements are independent and an observation sequence is denoted $O = (y_0, \cdots, y_t, \cdots)$, the likelihood function $p(O \mid \boldsymbol{A})$ is the same as equation (5.41) in the MRF-based prediction. The posterior distribution is

$$p(\boldsymbol{A} \mid O) = \frac{p(O \mid \boldsymbol{A}) p(\boldsymbol{A})}{p(O)}, \tag{5.63}$$

where $p(O)$ is the normalizing constant.

Similarly to the parameter estimation of the MRF-based prediction in the previous section, the EM algorithm is applied to estimate the parameters and its Q function is similar to the one in equation (5.45). With different prior knowledge from the MRF-based prediction in the previous section, the Q function is expanded as

$$\begin{aligned} Q(\boldsymbol{A}, \boldsymbol{A}^{(k)}) &= E_{\mathcal{M}|O,\boldsymbol{A}^{(k)}} \log p(\mathcal{M}, O \mid \boldsymbol{A}) + \log p(\boldsymbol{A}) \\ &= E_{\mathcal{M}|O,\boldsymbol{A}^{(k)}} \log p(O \mid \mathcal{M}, \boldsymbol{A}) + E_{\mathcal{M}|O,\boldsymbol{A}^{(k)}} \log p(\mathcal{M} \mid \boldsymbol{A}) \\ &\quad - 2\log\left(\sqrt{(2\pi)^n |K_{\mathcal{I}\mathcal{I}}|}\right) - \frac{1}{2} U(\boldsymbol{a}_{11}) - \frac{1}{2} U(\boldsymbol{a}_{22}). \end{aligned} \tag{5.64}$$

The probabilities $p(O \mid \mathcal{M}, \boldsymbol{A})$ can be calculated from the known sensor model, and the detail can be found in Section 5.2.1. The same parameter estimation can be obtained by maximizing the non-constant terms. Disregarding the first and third constant terms gives

$$\begin{aligned} & E_{\mathcal{M}|O,\boldsymbol{A}^{(k)}} \log p(\mathcal{M} \mid \boldsymbol{A}) - \frac{1}{2} U(\boldsymbol{a}_{11}) - \frac{1}{2} U(\boldsymbol{a}_{22}) \\ &= \sum_{c \in \mathcal{I}} f(\rho^c_1) + \sum_{c \in \mathcal{I}} f(a^c_{11}) + \sum_{c \in \mathcal{I}} f(a^c_{22}) - \frac{1}{2} U(\boldsymbol{a}_{11}) - \frac{1}{2} U(\boldsymbol{a}_{22}) \\ &= f(\boldsymbol{\rho}_1) + f(\boldsymbol{a}_{11}) + f(\boldsymbol{a}_{22}), \end{aligned} \tag{5.65}$$

where the function $f(\boldsymbol{\rho}_1)$ is the same as equation (5.48) and $\boldsymbol{\rho}_1$ is the vector of the initial occupancy probabilities. The functions $f(\boldsymbol{a}_{11})$ and $f(\boldsymbol{a}_{22})$, defined by

$$\begin{aligned} f(\boldsymbol{a}_{11}) &= \sum_{c \in \mathcal{I}} f(a^c_{11}) - \frac{1}{2} U(\boldsymbol{a}_{11}) \\ &= \boldsymbol{\xi}_{11} \log \boldsymbol{a}_{11} + \boldsymbol{\xi}_{12} \log (\boldsymbol{1} - \boldsymbol{a}_{11}) - \frac{1}{2} U(\boldsymbol{a}_{11}), \end{aligned} \tag{5.66}$$

$$f(\boldsymbol{a}_{22}) = \sum_{c\in\mathcal{I}} f(a_{22}^c) - \frac{1}{2}U(\boldsymbol{a}_{22})$$
$$= \boldsymbol{\xi}_{22}\log \boldsymbol{a}_{22} + \boldsymbol{\xi}_{21}\log(\mathbf{1}-\boldsymbol{a}_{22}) - \frac{1}{2}U(\boldsymbol{a}_{22}). \tag{5.67}$$

The vector $\boldsymbol{\xi}_{ij} = [\cdots, \sum_t \xi_{ij}^c(t), \cdots](c \in \mathcal{I})$, where $\xi_{ij}^c(t) = p(m_c^t = s_i, m_c^{t+1} = s_j \mid O, \theta_c^{(k)})$ which is the probability of being in state s_i in time instant t and s_j in time instant $t+1$ given the observation sequence O. The three functions contains different parameters and can be maximized individually. The estimation of ρ_1 is shown in equation (5.54). The derivatives of $Q(\boldsymbol{A}, \boldsymbol{A}^{(k)})$ with respect to $\boldsymbol{a}_{11}$ and $\boldsymbol{a}_{22}$ are given by

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{a}_{11}} f(\boldsymbol{a}_{11}) = \boldsymbol{\xi}_{11}^\top \oslash \boldsymbol{a}_{11} - \boldsymbol{\xi}_{12}^\top \oslash (\mathbf{1}-\boldsymbol{a}_{11}) - K_{\mathcal{I}\mathcal{I}}^{-1}(\log\frac{\boldsymbol{a}_{11}}{\mathbf{1}-\boldsymbol{a}_{11}} - \boldsymbol{\mu}_1) \odot \frac{\mathbf{1}}{\boldsymbol{a}_{11}\odot(\mathbf{1}-\boldsymbol{a}_{11})}, \tag{5.68}$$

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{a}_{22}} f(\boldsymbol{a}_{22}) = \boldsymbol{\xi}_{22}^\top \oslash \boldsymbol{a}_{22} - \boldsymbol{\xi}_{21}^\top \oslash (\mathbf{1}-\boldsymbol{a}_{22}) - K_{\mathcal{I}\mathcal{I}}^{-1}(\log\frac{\boldsymbol{a}_{22}}{\mathbf{1}-\boldsymbol{a}_{22}} - \boldsymbol{\mu}_2) \odot \frac{\mathbf{1}}{\boldsymbol{a}_{22}\odot(\mathbf{1}-\boldsymbol{a}_{22})}. \tag{5.69}$$

The divisions are also elementwise. However, the parameter vectors $\boldsymbol{a}_{11}$ and $\boldsymbol{a}_{22}$ can not be solved directly. Algorithm 5 is used to estimate them.

5.3.2 Hidden Markov model parameter prediction

The EM algorithm in the previous section does not give the variances of the parameters $\boldsymbol{a}_{11}$ and $\boldsymbol{a}_{22}$. After the HMM parameter estimation, all the noise in the observations are assumed to be filtered out and the estimates of $\boldsymbol{a}_{11}$ and $\boldsymbol{a}_{22}$ are assumed to be without noise. Because of the independence between a_{11}^c and a_{22}^c, the prediction problem can be divided into two individual subtasks,

$$p(\boldsymbol{A}_* \mid \boldsymbol{A}) = p(\boldsymbol{a}_{11}^* \mid \boldsymbol{a}_{11}) p(\boldsymbol{a}_{22}^* \mid \boldsymbol{a}_{22}). \tag{5.70}$$

The two parameters of the test points can be predicted individually. Assuming the log odds forms of parameter vectors $\boldsymbol{a}_{11}^*$ and $\boldsymbol{a}_{22}^*$ are denoted by $\boldsymbol{l}_{a_{11}}^* = [\cdots, l_{a_{11}}^c, \cdots]^\top (c \in \mathcal{I}_*)$ and $\boldsymbol{l}_{a_{22}}^* = [\cdots, l_{a_{22}}^c, \cdots]^\top (c \in \mathcal{I}_*)$, the estimations $p(\boldsymbol{a}_{11}^* \mid \boldsymbol{a}_{11})$ and $p(\boldsymbol{a}_{22}^* \mid \boldsymbol{a}_{22})$ can be derived from $p(\boldsymbol{l}_{a_{11}}^* \mid \boldsymbol{l}_{a_{11}})$ and $p(\boldsymbol{l}_{a_{22}}^* \mid \boldsymbol{l}_{a_{22}})$, respectively. The joint distribution of $\boldsymbol{l}_{a_{11}}$ and $\boldsymbol{l}_{a_{11}}^*$ is

$$\begin{bmatrix} \boldsymbol{l}_{a_{11}} \\ \boldsymbol{l}_{a_{11}}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_1^* \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}_{\mathcal{I}\mathcal{I}} & \boldsymbol{K}_{\mathcal{I}*}^\top \\ \boldsymbol{K}_{\mathcal{I}*} & \boldsymbol{K}_{**} \end{bmatrix} \right), \tag{5.71}$$

where $\boldsymbol{\mu}_1^*$ is the mean vector of $\boldsymbol{l}_{a_{11}}^*$, the matrix $\boldsymbol{K}_{\mathcal{I}*}$ denotes the covariance between $\boldsymbol{l}_{a_{11}}$ and $\boldsymbol{l}_{a_{11}}^*$, and $\boldsymbol{K}_{**}$ is the covariance matrix of $\boldsymbol{l}_{a_{11}}^*$. The predictive distribution with noise-free observations is

$$\boldsymbol{l}_{a_{11}}^* \mid \boldsymbol{l}_{a_{11}} \sim \mathcal{N}(\bar{\boldsymbol{l}}_{a_{11}}^*, \hat{\boldsymbol{K}}_a^*), \tag{5.72}$$

where the predictive mean vector $\bar{\boldsymbol{l}}_{a_{11}}^*$ and covariance matrix $\hat{\boldsymbol{K}}_a^*$ are given by

$$\bar{\boldsymbol{l}}_{a_{11}}^* = \boldsymbol{\mu}_1^* + \boldsymbol{K}_{\mathcal{I}*}\boldsymbol{K}_{\mathcal{I}\mathcal{I}}^{-1}(\boldsymbol{l}_{a_{11}} - \boldsymbol{\mu}_1), \tag{5.73}$$

$$\hat{\boldsymbol{K}}_a^* = \boldsymbol{K}_{**} - \boldsymbol{K}_{\mathcal{I}*}\boldsymbol{K}_{\mathcal{I}\mathcal{I}}^{-1}\boldsymbol{K}_{\mathcal{I}*}^\top. \tag{5.74}$$

The best predictive parameter vector $\boldsymbol{a}_{11}^*$ of the test points is given by the logistic function

$$\boldsymbol{a}_{11}^* = \frac{\mathbf{1}}{\mathbf{1} + \exp\left(-\bar{\boldsymbol{l}}_{a_{11}}^*\right)}, \tag{5.75}$$

where $\mathbf{1}$ is a column of ones and the division is elementwise. Similarly, the joint distribution of $\boldsymbol{l}_{a_{22}}$ and $\boldsymbol{l}_{a_{22}}^*$ is

$$\begin{bmatrix} \boldsymbol{l}_{a_{22}} \\ \boldsymbol{l}_{a_{22}}^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_2 \\ \boldsymbol{\mu}_2^* \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}_{\mathcal{I}\mathcal{I}} & \boldsymbol{K}_{\mathcal{I}*}^\top \\ \boldsymbol{K}_{\mathcal{I}*} & \boldsymbol{K}_{**} \end{bmatrix} \right), \tag{5.76}$$

where $\boldsymbol{\mu}_2^*$ is the mean vector of $\boldsymbol{l}_{a_{22}}^*$. The coordinates of training points and test points for two prediction problems are the same. As a result, the covariance matrix of the joint distribution and the predictive covariance matrix do not change. The predictive distribution of $\boldsymbol{l}_{a_{22}}^*$ is

$$\boldsymbol{l}_{a_{22}}^* \mid \boldsymbol{l}_{a_{22}} \sim \mathcal{N}(\bar{\boldsymbol{l}}_{a_{22}}^*, \hat{\boldsymbol{K}}_a^*), \tag{5.77}$$

where the predictive mean vector $\bar{\boldsymbol{l}}_{a_{22}}^*$ is given by

$$\bar{\boldsymbol{l}}_{a_{22}}^* = \boldsymbol{\mu}_2^* + \boldsymbol{K}_{\mathcal{I}*}\boldsymbol{K}_{\mathcal{I}\mathcal{I}}^{-1}(\boldsymbol{l}_{a_{22}} - \boldsymbol{\mu}_2). \tag{5.78}$$

The best predictive parameter vector $\boldsymbol{a}_{22}^*$ of the test points is

$$\boldsymbol{a}_{22}^* = \frac{\mathbf{1}}{\mathbf{1} + \exp\left(-\bar{\boldsymbol{l}}_{a_{22}}^*\right)}. \tag{5.79}$$

With the Ornstein–Uhlenbeck kernel function, one point is correlated mostly with points in a small neighbourhood and the prediction can be implemented using only local information.

5.3.3 Computational complexity

The prediction based on GRFs in dynamic environments is described as the Algorithm 6 below.

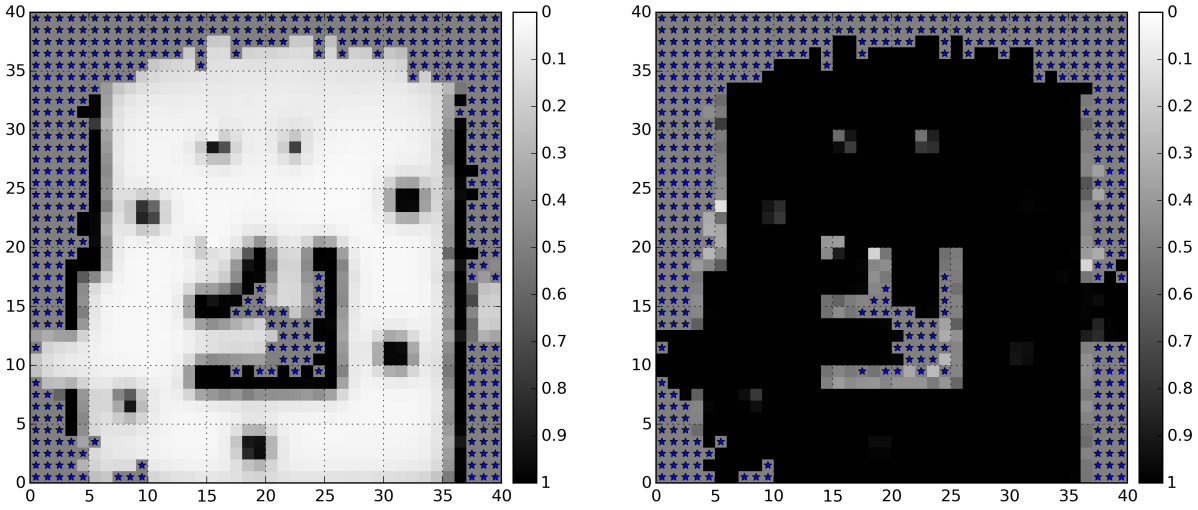
Algorithm 6 Prediction based on GRFs in dynamic environments

Input: Measurements z_i^t

Output: $\boldsymbol{a}_{11}^{(k)}$, $\boldsymbol{a}_{22}^{(k)}$, $\boldsymbol{a}_{11}^*$ and $\boldsymbol{a}_{22}^*$

Initialize $step$ $\boldsymbol{\rho}_1$, $\boldsymbol{a}_{11}$, $\boldsymbol{a}_{22}$ and $k = 0$
Compute the inverse matrix $\boldsymbol{K}_{\mathcal{I}\mathcal{I}}^{-1}$
Build occupancy grid maps with temporary state probabilities 0.5 and obtain $p(m_c^t \mid y_t)$
while $k <$ maximum iteration **do**
    Calculate $\boldsymbol{\gamma}_1$ and $\boldsymbol{\xi}_{ij}$ using equations (5.22) and (5.23)
    $\boldsymbol{\rho}_1^{(k+1)} = \boldsymbol{\gamma}_1$
    $\boldsymbol{a}_{11}^{(k+1)}$ =LSM$(-f(\boldsymbol{a}_{11}), \boldsymbol{a}_{11}^{(k)}, step)$
    $\boldsymbol{a}_{22}^{(k+1)}$ =LSM$(-f(\boldsymbol{a}_{22}), \boldsymbol{a}_{22}^{(k)}, step)$
    $k = k + 1$
Obtain the parameters $\boldsymbol{a}_{11}^*$ and $\boldsymbol{a}_{22}^*$ using equations (5.75), (5.73), (5.79) and (5.78).

The computational complexity is obtained as follows.



(a) a_{11}: staying in occupied state

(b) a_{22}: staying in free state

Figure 5.16: HMM parameter estimation of the GRF-based prediction for observed space in the simulated dynamic environment

Step 2 Inverting matrix $\boldsymbol{K}_{\mathcal{I}\mathcal{I}}$ has a computational complexity of $\mathcal{O}(n^3)$, where n is the number of training points.

Step 3 The problems with different numbers of parameters have different maximum iterations.

Step 4 As described in Algorithm 4, the computational complexity of this step is $\mathcal{O}(\zeta n)$, where ζ is the size of the observation sequence O.

Step 5 The complexity of this step is $\mathcal{O}(\zeta n)$.

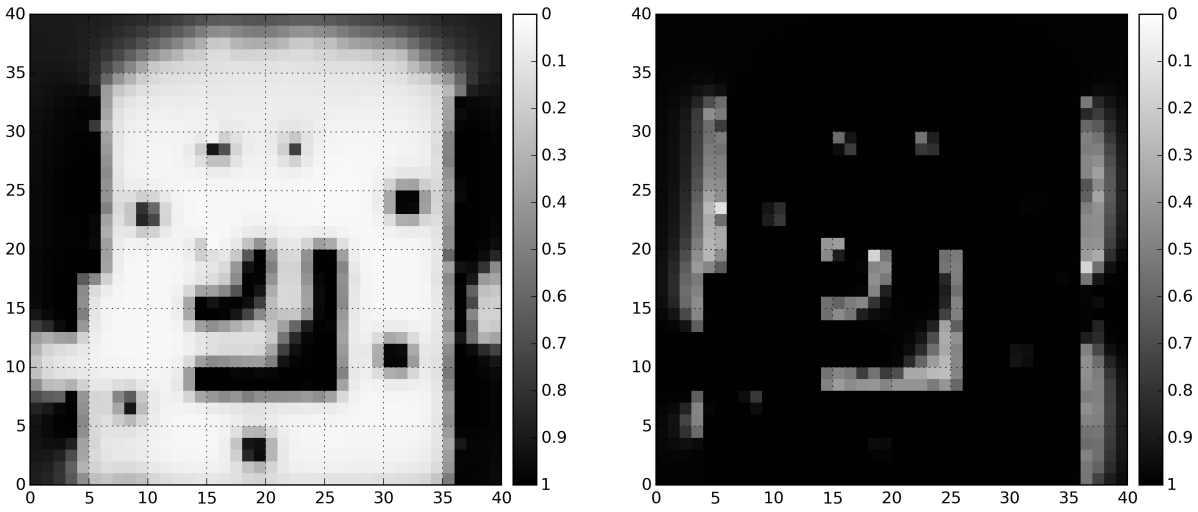
Steps 7 and 8 After factorization, there are fewer parameters in these two steps and their computational complexities are the same as $\mathcal{O}(n^2)$.

Step 10 The parameters $\boldsymbol{a}_{11}^*$ and $\boldsymbol{a}_{22}^*$ are obtained based on the prediction equation. The computational complexity is $\mathcal{O}(n_t n^2)$, where n_t denotes the number of test points.

5.3.4 Simulation

The simulated observation data in Section 5.1 is also used here. The training points are the central points of the grid cells in observed space and the test points are only chosen from unobserved space. The initial values of parameters are 0.5, the length-scale $\ell = 3$, and the signal variance $\sigma_f^2 = 25$. The mean vectors $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_1^*$ are set to $\log 9$, which corresponds to a probability of 0.9. The mean vectors $\boldsymbol{\mu}_2$ and $\boldsymbol{\mu}_2^*$ are set to $\log 99$ which corresponds to a probability of 0.99. It means the prior knowledge of the environment is slow dynamic. The maximum number of iterations of the optimization process is set to 2000. The parameter estimation for observed space is shown in Figure 5.16, which is similar to Figure 5.14. Because the means in the prior distribution are set to high values, the estimates are higher than those in Figure 5.14, and the estimates of the walls in Figure 5.16(b) are close to 0.5.

Based on the estimation of observed space, the estimation of unobserved space is shown in Figure 5.17. The unknown space close to the walls is predicted with a high a_{11} and a low a_{22}. The unknown space close



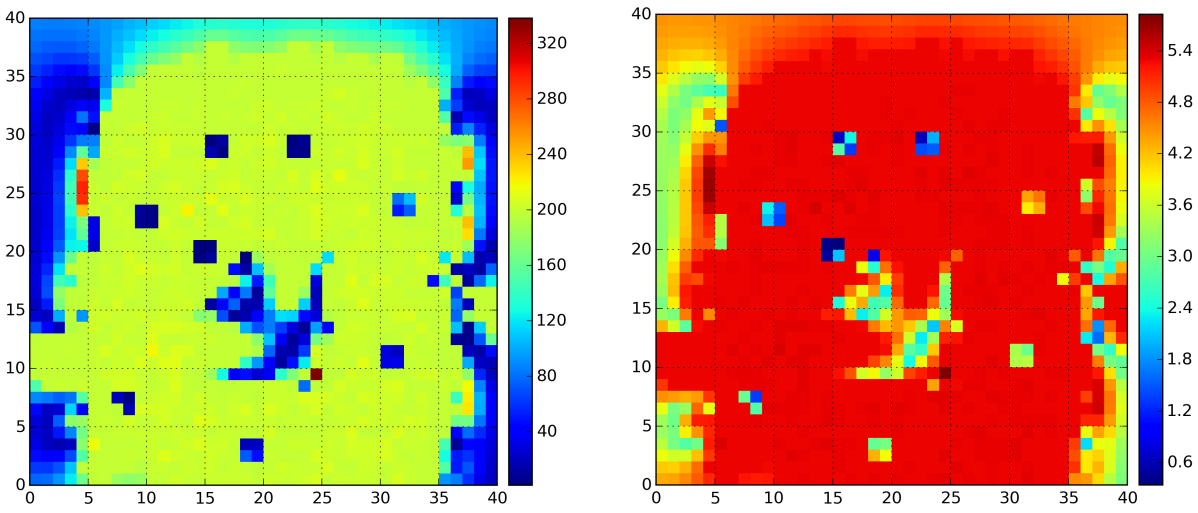
(a) a_{11}: staying in occupied state

(b) a_{22}: staying in free state

Figure 5.17: HMM parameter prediction of the GRF-based prediction for unobserved space in the simulated dynamic environment

to free space is predicted with a low a_{11} and a high a_{22}. The parameters of the unobserved space a little far from the observed space are the same as the prior values.

The overall expected duration is shown in Figure 5.18. There is no much difference with that of the MRF-based prediction in observed space. Static space and slow dynamic objects have long expected durations. The unobserved space close to the free space and walls has a long expected duration.



(a) Overall expected duration

(b) Overall expected duration in log scale

Figure 5.18: Overall expected duration of the GRF-based prediction for the simulated dynamic environment

5.4 Comparison

The HMM parameters for each grid cell without prior knowledge can be estimated individually, and it is easy to obtain the best estimates. With an MRF prior, the parameter estimate is smoother. However, every parameter depends on the others, and it is not easy to search for the best estimates of the free space without occupied observations and of the occupied space without free observations. With a Gaussian prior, the prediction is factorized into two subtasks: parameter estimation for training points and parameter prediction for test points. The parameter estimation is similar to the MRF-based prediction. However, there are fewer parameters in this step, and it reduces the computational cost. In the prediction step, the optimal solution is easy to be obtained given the parameter estimation in the previous step.

Based on parameters a_{11} and a_{22}, the space in dynamic environments can be classified as Table 5.4. The objects 1, 2, 3, 4 and 5 in the simulated map are high dynamic, and the other objects and the door are low dynamic. The classification results for Figures 5.5, 5.14 and 5.17 are shown as Figure 5.19 and Table 5.5. These figures are the results of independent assumption and the proposed methods. Because the space with fewer observations is not optimized in the MRF-based prediction, the true free (TF) space decreases and the false high dynamic (FHD) space increases. Meanwhile, it filters out false low dynamic (FLD) space. Smoothing the map also causes the increase of FHD and the GRF-based prediction has the same problem. However, the GRF-based method predicts more TF. Because the prior knowledge of the map is low dynamic and the space behind the walls are predicted occupied, there are more FLD and false occupied (FO) space.

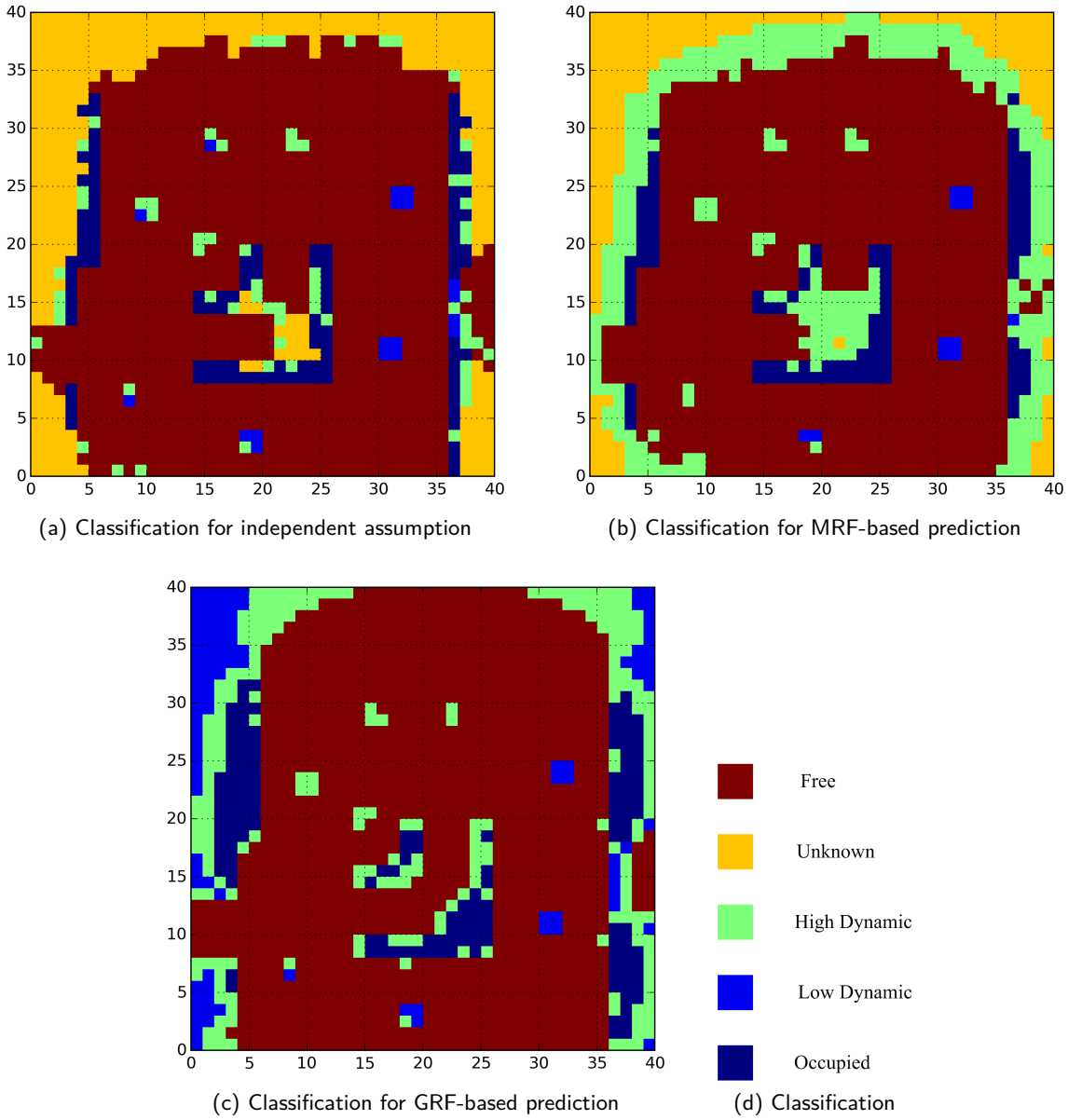
Table 5.4: Classification for the dynamic environments. When the space does not belong to any class in the previous four, it is classified as high dynamic.

Classification	Free	Occupied	Low Dynamic	Unknown	High Dynamic
Parameters	$a_{11} < 0.6$ $a_{22} > 0.85$	$a_{11} > 0.85$ $a_{22} < 0.6$	$a_{11} > 0.85$ $a_{22} > 0.85$	$0.53 > a_{11} > 0.47$ $0.53 > a_{22} > 0.47$	Others

Table 5.5: Comparison between results for the simulated dynamic map. TF = True free, TO = True occupied, FF = False free, FO = False occupied, TLD = True low dynamic, FLD = False low dynamic, THD = True high dynamic, FHD = False high dynamic, UN = Unknown.

Classification	TF	FF	TO	FO	TLD	FLD	THD	FHD	UN
Independent assumption (Figure 5.5)	1061	5	97	15	15	3	12	54	338
MRF-based prediction (Figure 5.14)	974	6	90	20	11	0	15	292	192
GRF-based prediction (Figure 5.17)	1126	8	72	75	16	85	13	205	0

The HMM parameter estimation without prior knowledge is similar to the work reviewed in Section 2.4.2. In [MDBB12], the backward procedure is discarded, and an online method is applied to estimate the HMM parameters. The Baum–Welch algorithm can obtain a better estimate than the online method with the same data. The disadvantage is that it cannot be done online. Based on the input-output HMM [BF95], the current state of one grid cell also depends on the previous observations of neighbouring grid cells in [WAJF14]. In [DKS15], the hidden semi-Markov model is applied to incorporate state duration. Different from these methods, the proposed prediction methods consider the dependence between the transition probabilities, which can smooth the parameter estimate of observed space and estimate the parameters of unobserved space. The main disadvantage of the proposed methods is that the estimation for the border of observed space has very slow convergence.



(a) Classification for independent assumption

(b) Classification for MRF-based prediction

(c) Classification for GRF-based prediction

(d) Classification

Figure 5.19: Classification of the results for the simulated dynamic environment

5.5 Summary

In this chapter, Markov chains are applied to model dynamic environments. Occupancy grid mapping is used to build maps for every time instant, and the posterior probabilities are used to estimate the parameters directly instead of the emission probabilities. Firstly, the transition matrix of each grid cell is estimated individually without prior knowledge. Furthermore, the MRF is applied to consider the parameter dependence between different grid cells. With the MRF-based prior knowledge, the optimum in the maximizing step of the EM algorithm cannot be obtained directly, and a line search method is applied. Finally, the prediction based on GRFs is factorized to reduce computational complexity. Simulations are done to test these methods with the same observation data.

6

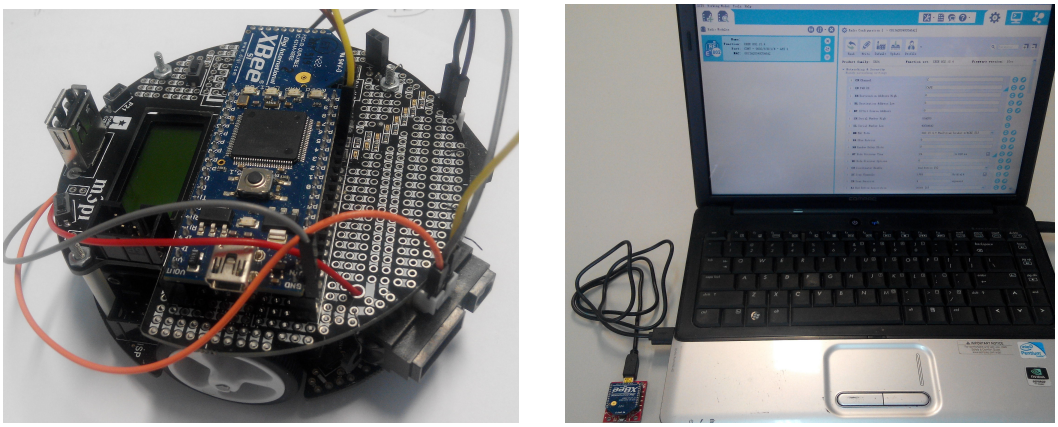
Experiments

In the previous two chapters, different methods are proposed for static and dynamic environments individually. In this chapter, these methods are validated in an experimental setup. The experimental platform is introduced in Section 6.1. The uncertainty of robot pose is estimated in Section 6.2. Section 6.3 shows how the uncertainty of the robot pose can be incorporated in the proposed methods. The results of the proposed methods for static and dynamic environments are shown in Sections 6.4 and 6.5, respectively.

6.1 Experimental setup

As shown in Figure 6.1, the experimental platform consists of two parts: the robot part and the PC part. On the robot part, a 3pi robot and one XBee communication module are connected to an mbed expansion board, two IR sensors are connected to the mbed. On the PC part, one XBee module is connected to the PC by an XBee Explorer USB.

The block diagram of the platform is shown in Figure 6.2. The 3pi robot is controlled by the mbed microprocessor which sends commands to the robot by a pair of serial ports. The IR sensors 1 and 2 are



(a) Robot part (b) PC part

Figure 6.1: Experiment platform

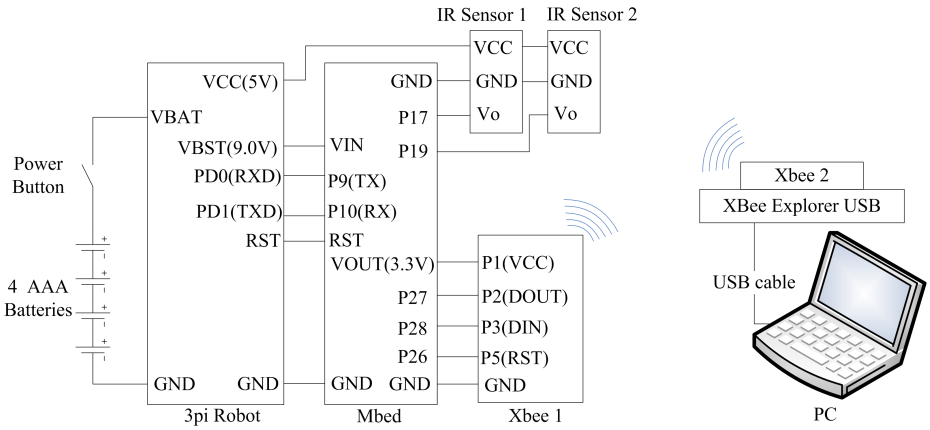


Figure 6.2: Block diagram of experiment platform

two Sharp GP2Y0A41SK0F IR sensors, which can measure distances to objects and generate an analog voltage signal. The mbed samples the analog voltages of the IR sensors by two ADC ports. The voltage data can be sent to the PC by XBees 1 and 2, which are XBee S1 802.15.4 low-power modules. The mbed sends data to XBee 1 by another pair of serial ports. XBee 2 receives the data from XBee 1 and sends it to the PC. Since the mbed processor has limited computational power, the PC is in charge of the mapping tasks.

6.1.1 Software tools

The experimental platform requires software to be installed and configured on the PC, the mbed microprocessor and the 3pi microcontroller. The following sections describe the software used and its configuration for each of these platforms.

The 3pi robot

The 3pi robot operates as a slave of the mbed. A slave program is provided in the Pololu AVR C/C++ Library which can be found on the website of Pololu robotics and electronics. The main function of this slave program is to analyse and follow the commands sent by the mbed. The commands include controlling the speeds of the wheels, calibrating QTR-RC reflectance sensors, returning the feedback of QTR-RC reflectance sensors, and so on. The slave program can be compiled by the recommended development platforms without change. In the experiments, the slave program is compiled with Atmel Studio 7 in Windows 10. A HEX file is generated which can be loaded on the 3pi robot microcontroller. An external USB AVR programmer is required to program the 3pi robot. The driver of the programmer can also be downloaded on the website of Pololu robotics and electronics. The programmer is connected to the 3pi robot by a 6-pin ISP programming cable and the PC by a USB A to mini-B cable.

XBee modules

The XBee modules are configured using the XCTU software tool. It requires the driver of the USB-based XBee explorer to be installed. The chip on the explorer is a FTDI FT231X whose driver can be downloaded on the website of FTDI. After installing the driver, the explorer is assigned a port number. Some libraries for different programming languages, such as XBee Python Library, XBee Java library, are available. In the experiments, the XBee Python library is installed. Python 3.0+ and PySerial 3.0+ are required by the XBee Python library. In the experiments, Python 3.4 and PySerial 3.01 are installed.

The mbed

The C++ compiler and integrated development environment (IDE) of the mbed are free and web-based. The common browsers on a PC can be used for programming. The program is compiled remotely on the web server, which provides the binary for download. This binary can then be uploaded to the mbed using the USB A to mini-B cable. There is a built-in USB flash programmer on the mbed, and no external programmer is required. The basic library for the mbed is the C++ SDK. The AnalogIn interface is used to sample the voltages of the IR sensors and the timer interface is used to trigger an interrupt function. Even though this library contains the basic interfaces to communicate with the 3pi robot and the XBee 1, it is easier to use other two libraries: an m3pi library and an XBee library. These libraries can be imported to mbed projects in the web-based IDE.

6.1.2 The mbed

The mbed is a high-level controller from ARM, and its core is a high-performance NXP LPC1768 Cortex-M3 processor. It runs at 96 MHz with 32 KiB RAM and 512 KiB FLASH. It includes several peripherals and interfaces, such as Ethernet, 2 $\times$ I2C, 3 $\times$ UART, 6 $\times$ PWM, 6 $\times$ 12-bit ADC. In the experiments, two ADCs and two UARTs are used. The two ADCs can be any two of the six, and their voltage inputs should not exceed 3.3 V. The two UARTs are connected to XBee 1 and the 3pi robot through the expansion board. The power supply of the mbed is 5V that could be provided by USB while programming the board or the 3pi robot. While the robot is running, the mbed is powered by four AAA batteries on the 3pi robot.

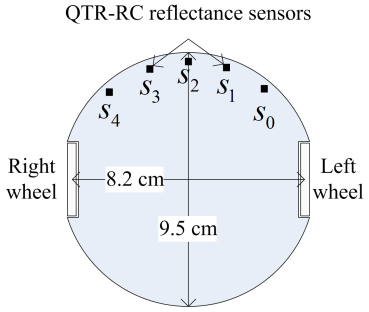


Figure 6.3: Brief bottom view of the robot

6.1.3 The robot

The brief bottom view of the robot is shown in Figure 6.3. The width W between the two wheels is 8.2 cm, and the outer diameter of the robot is 9.5 cm. In the front, there are five QTR-RC reflectance sensors which can be used to follow black lines. These sensors are connected to the digital pins. Despite being digital pins, a scheme based on the discharge rate of pull-up capacitors allows analog reflectance measurements to be taken for each of the five sensors. The provided feedback function in the slave program combines all the feedbacks of five sensors, and the integrated feedback is computed by

$$\frac{1000s_1 + 2000s_2 + 3000s_3 + 4000s_4}{s_0 + s_1 + s_2 + s_3 + s_4}. \tag{6.1}$$

The range of the feedback is [0, 4000]. The speeds of two wheels can be changed individually by the provided motor function which accepts a speed parameter with range [-255, +255]. The number encodes the speed level, and the sign is the direction of the motor. If the sign is positive, the motor will move forward. The maximum speed of the robot is 100 cm/s. However, the speed does not change linearly with command.

6.1.4 The IR sensors

The two IR sensors are mounted on the robot as shown in Figure 6.4. The relative orientations are $\pm 30^\circ$ with respect to the robot's reference frame. The measuring range is 4 to 30 cm, and the update period is 16.5 $\pm$ 4 ms. The operating voltage is 4.5 to 5.5 V, and connected to the VCC(5.0 V) port of the 3pi robot. The output voltage is not more than 3.3 V which is the maximum voltage input of the ADC ports on the mbed. When the distance is more than 20 cm, the output voltage has lower sensitivity and becomes noisier. In the experiments, the maximum distances of IR sensors are set to 20 cm. The tested relationship between measured distances and output voltages is shown in Figure 6.5. Since the aperture angle is very small, the two IR sensors do not interface with each other, and the beam model of range finders in Section 4.1 can be applied to these IR sensors.

6.1.5 The XBee modules

The XBees 1 and 2 are of the same type, and they can communicate with each other without special configuration. In order to use the mbed Library and XBee Python Library provided by Digi, these two XBees should have the same "ID" and networking settings. The sleep mode "SM" setting is set to "0 = No Sleep", and the API mode "AP" must be set to 1 or 2. In order to simplify the setting step, the default

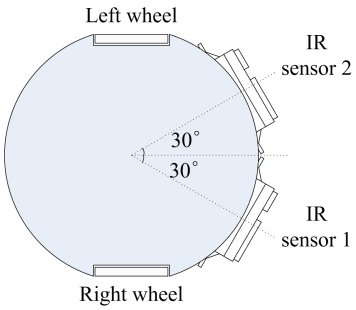


Figure 6.4: Brief top view of the robot

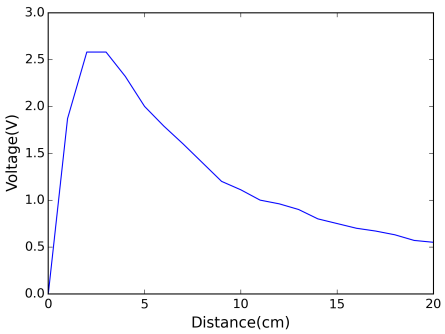
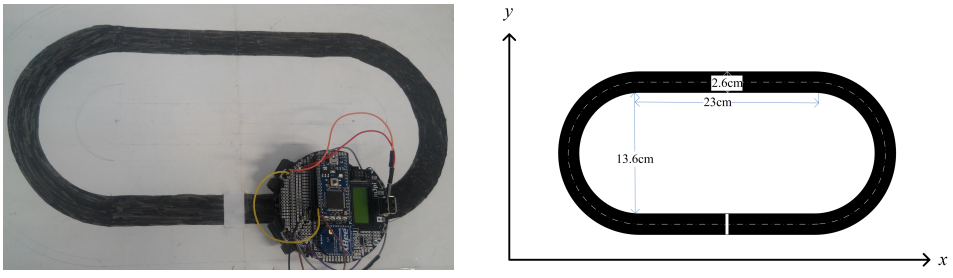


Figure 6.5: Tested relationship between the measured distance and the output voltage

settings are used except the "AP" setting which is configured once.

6.2 Localization

The proposed methods assume known poses. However, the robot cannot know its pose precisely in real environments. All the poses are corrected by a trajectory closure constraint. In order to help the robot to close its trajectory, a track with a mark is drawn as in Figure 6.6(a) and its size is shown in Figure 6.6(b). A simple controller (see Appendix D) is designed to help the robot follow the track. At the beginning, the robot starts at the mark. As the robot moves, its pose is predicted by its motion model while the corresponding uncertainty increases. When the robot detects the mark again, the robot closes its trajectory, and all the poses will be corrected.



(a) Track (b) Size of the track

Figure 6.6: Track and its size. The black part is the track and the white part on the track is a mark.

6.2.1 Robot motion model

The robot position is represented by the central point between two wheels in world coordinates, and its orientation is relative to the x axis. Its pose vector is denoted by $\boldsymbol{\mathcal{P}} = [x, y, \phi]^\mathsf{T}$, which includes the position (x, y) and the orientation ϕ. Assuming the speeds of the left and right wheels are denoted by v_l and v_r, the speed of the robot is modelled as

$$\begin{cases} \frac{\mathrm{d}}{\mathrm{d}t}x = \frac{v_l + v_r}{2}\cos\phi \\ \frac{\mathrm{d}}{\mathrm{d}t}y = \frac{v_l + v_r}{2}\sin\phi \\ \frac{\mathrm{d}}{\mathrm{d}t}\phi = \frac{v_r - v_l}{W}. \end{cases} \tag{6.2}$$

When the robot goes straight shown in Figure 6.7(a), its orientation does not change. Assuming the pose at time step $k-1$ is denoted by $\boldsymbol{\mathcal{P}}_{k-1} = (x_{k-1}, y_{k-1}, \phi_{k-1})$, the next pose $\boldsymbol{\mathcal{P}}_k$ after a time interval Δt can be computed based on the Euler method,

$$\begin{cases} x_k = x_{k-1} + \frac{v_l + v_r}{2}\Delta t\cos\phi_{k-1} \\ y_k = y_{k-1} + \frac{v_l + v_r}{2}\Delta t\sin\phi_{k-1} \\ \phi_k = \phi_{k-1}. \end{cases} \tag{6.3}$$

When the robot does not move straight shown in Figure 6.7(b), a direct intergration is performed instead of the Euler approximation. Integrating equation (6.2) in a time interval Δt gives the new pose [TBF05]

$$\begin{cases} x_k = x_{k-1} + \frac{v_l + v_r}{2(v_r - v_l)}\left(\sin\left(\phi_{k-1} + \frac{v_r - v_l}{W}\Delta t\right) - \sin\phi_{k-1}\right) \\ y_k = y_{k-1} + \frac{v_l + v_r}{2(v_r - v_l)}\left(-\cos\left(\phi_{k-1} + \frac{v_r - v_l}{W}\Delta t\right) + \cos\phi_{k-1}\right) \\ \phi_k = \phi_{k-1} + \frac{v_r - v_l}{W}\Delta t. \end{cases} \tag{6.4}$$

The system model with additive process noise $\mathcal{W}$ is rewritten as

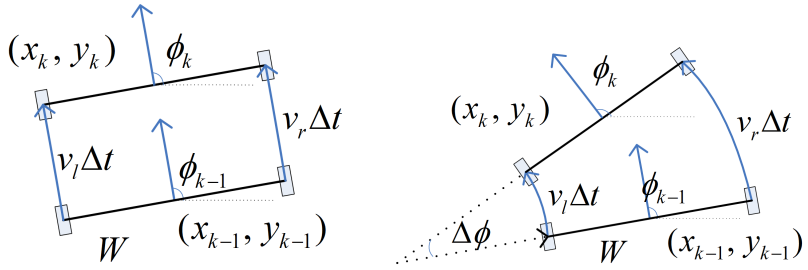
$$\boldsymbol{\mathcal{P}}_k = F(\boldsymbol{\mathcal{P}}_{k-1}, v_r, v_l) + \mathcal{W}, \tag{6.5}$$

where $\mathcal{W}$ is Gaussian distributed with zero mean and covariance $\boldsymbol{\mathcal{R}}$. The function $F(\boldsymbol{\mathcal{P}}_{k-1}, v_r, v_l)$ represents the expressions on the right sides of equations (6.3) and (6.4).

6.2.2 Robot measurement model

The robot follows the track in clockwise direction. Once the robot detects the mark, there will be one observation of the position of the robot. The part of the track in front of the mark is straight. When the robot arrives at the mark, the simple controller can adjust the orientation of the robot to be close to -π. As a result, not only the position but also the orientation becomes known, although not precisely. However, the observation is not precise. The noisy observation is modelled as

$$\boldsymbol{\mathcal{Z}}_k = \boldsymbol{\mathcal{P}}_k + [-4.75, 0, 0]^\mathsf{T} + \mathcal{V}, \tag{6.6}$$



(a) going straight (b) turning

Figure 6.7: Robot motion model. Two wheels are represented by rectangles connected by a black line.

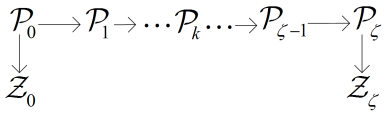


Figure 6.8: Two observations for the robot

where $\mathcal{V}$ is Gaussian noise with zero mean and covariance $\boldsymbol{\mathcal{Q}}$. The measurement $\boldsymbol{\mathcal{Z}}_k$ is the pose of the robot head where the reflectance sensors are located and $\boldsymbol{\mathcal{P}}_k$ is the pose of the central point of the robot. The pose difference between the head and the central point is $[-4.75, 0, 0]^\mathsf{T}$ when the mark is detected.

6.2.3 Pose smoothing

When the robot starts at the mark and performs a complete loop returning to the mark, only two observations of the pose are available, as shown in Figure 6.8. The two observations are denoted by $\boldsymbol{\mathcal{Z}}_0$ and $\boldsymbol{\mathcal{Z}}_\zeta$, respectively. Assuming the position of the mark is at coordinates $(x_0,\ y_0)$, both observations $\boldsymbol{\mathcal{Z}}_0$ and $\boldsymbol{\mathcal{Z}}_\zeta$ are equal to $(x_0,\ y_0,\ \text{-}\pi)$. In order to ensure the smoothness of pose estimation along the trajectory, all the poses along the trajectory should be corrected by the two observations. The pose smoothing can be obtained by

$$p(\boldsymbol{\mathcal{P}}_k \mid \boldsymbol{\mathcal{Z}}_0, \boldsymbol{\mathcal{Z}}_\zeta) = \frac{p(\boldsymbol{\mathcal{Z}}_\zeta \mid \boldsymbol{\mathcal{P}}_k) p(\boldsymbol{\mathcal{P}}_k \mid \boldsymbol{\mathcal{Z}}_0)}{p(\boldsymbol{\mathcal{Z}}_\zeta \mid \boldsymbol{\mathcal{Z}}_0)}. \tag{6.7}$$

This formula is divided into two steps: a forward step $p(\boldsymbol{\mathcal{P}}_k \mid \boldsymbol{\mathcal{Z}}_0)$ and a backward step $p(\boldsymbol{\mathcal{Z}}_\zeta \mid \boldsymbol{\mathcal{P}}_k)$.

The forward step estimates $p(\boldsymbol{\mathcal{P}}_k \mid \boldsymbol{\mathcal{Z}}_0)$, which in general can be obtained by iterating the equation

$$p(\boldsymbol{\mathcal{P}}_k \mid \boldsymbol{\mathcal{Z}}_0) = \int p(\boldsymbol{\mathcal{P}}_k \mid \boldsymbol{\mathcal{P}}_{k-1}) p(\boldsymbol{\mathcal{P}}_{k-1} \mid \boldsymbol{\mathcal{Z}}_0) \mathrm{d}\boldsymbol{\mathcal{P}}_{k-1}. \tag{6.8}$$

Based on the previous predicted state distribution $p(\boldsymbol{\mathcal{P}}_{k-1} \mid \boldsymbol{\mathcal{Z}}_0)$, the current predicted state distribution $p(\boldsymbol{\mathcal{P}}_k \mid \boldsymbol{\mathcal{Z}}_0)$ can be obtained. Since the robot model is nonlinear and equation (6.8) is difficult to integrate, the forward step is done instead by the scaled unscented transformation as mentioned in Section 3.3.1. Because there is no prior knowledge for $\boldsymbol{\mathcal{P}}_0$, the posterior distribution $p(\boldsymbol{\mathcal{P}}_0 \mid \boldsymbol{\mathcal{Z}}_0)$ is assumed to be Gaussian distributed with mean $(x_0,\ y_0,\ \text{-}\pi)$ and covariance $\boldsymbol{\mathcal{Q}}$. The dimension L of the robot pose vector is 3, then the unscented Kalman filter mandates that $2L + 1$ sigma points should be sampled from the distribution $p(\boldsymbol{\mathcal{P}}_0 \mid \boldsymbol{\mathcal{Z}}_0)$. Assuming the sigma points are denoted by $\boldsymbol{\mathcal{P}}_0^i$, the corresponding mean weights w_m^i and variance weights w_c^i can be computed by equation (3.45) to approximate $p(\boldsymbol{\mathcal{P}}_0 \mid \boldsymbol{\mathcal{Z}}_0)$. Computing the mean and covariance weights requires three parameters α, β, and κ. Section 3.3.1 shows how them are chosen. Based on these sigma points, the other poses can be predicted by projecting sigma points through

the motion model step by step. After one prediction, new sigma points will be obtained, and they should be augmented as shown in equation (3.51) to include the system noise and predict the next pose. Assuming the augmented sigma points for pose $\boldsymbol{\mathcal{P}}_k$ are denoted by $\boldsymbol{\mathcal{P}}_k^i$, the mean vector and covariance matrix of the distribution $p(\boldsymbol{\mathcal{P}}_k \mid \boldsymbol{\mathcal{Z}}_0)$ can be approximated respectively by

$$\bar{\boldsymbol{\mathcal{P}}}_k^- \approx \sum_{i=0}^{2L} w_m^i \boldsymbol{\mathcal{P}}_k^i, \tag{6.9}$$

$$\boldsymbol{P}_k^- \approx \sum_{i=0}^{2L} w_c^i (\boldsymbol{\mathcal{P}}_k^i - \bar{\boldsymbol{\mathcal{P}}}_k^-)(\boldsymbol{\mathcal{P}}_k^i - \bar{\boldsymbol{\mathcal{P}}}_k^-)^\top. \tag{6.10}$$

The objective of the backward step is to estimate $p(\boldsymbol{\mathcal{Z}}_\zeta \mid \boldsymbol{\mathcal{P}}_k)$, which in general is given by iterating the equation

$$p(\boldsymbol{\mathcal{Z}}_\zeta \mid \boldsymbol{\mathcal{P}}_k) = \int p(\boldsymbol{\mathcal{Z}}_\zeta \mid \boldsymbol{\mathcal{P}}_{k+1}) p(\boldsymbol{\mathcal{P}}_{k+1} \mid \boldsymbol{\mathcal{P}}_k) \mathrm{d}\boldsymbol{\mathcal{P}}_{k+1}. \tag{6.11}$$

This equation means $p(\boldsymbol{\mathcal{Z}}_\zeta \mid \boldsymbol{\mathcal{P}}_k)$ can also be obtained recursively based on the robot motion model.

As in the forward step, equation (6.11) is difficult to integrate due to the nonlinear characteristics of the motion model, and the same approach based on the unscented transformation is used. The sigma points for the predicted observation are given by

$$\boldsymbol{z}_\zeta^i = \boldsymbol{\mathcal{P}}_\zeta^i + [-4.75, 0, 0]^\mathsf{T}. \tag{6.12}$$

Based on equations (3.63) and (3.64), the mean vector and covariance matrix of the predicted measurement are approximated respectively by

$$\bar{\boldsymbol{z}}_\zeta \approx \bar{\boldsymbol{\mathcal{P}}}_\zeta^- + [-4.75, 0, 0]^\mathsf{T}, \tag{6.13}$$

$$\boldsymbol{P}_z \approx \boldsymbol{P}_\zeta^- + \boldsymbol{\mathcal{Q}}. \tag{6.14}$$

For the pose $\boldsymbol{\mathcal{P}}_k$, every sigma point $\boldsymbol{\mathcal{P}}_k^i$ corresponds to a sigma point $\boldsymbol{z}_\zeta^i$ at the end of the time horizon. The corrected sigma points $\boldsymbol{\mathcal{P}}_k^{i+}$ used to approximate the corrected distribution $p(\boldsymbol{\mathcal{P}}_k \mid \boldsymbol{\mathcal{Z}}_0, \boldsymbol{\mathcal{Z}}_\zeta)$, which already include the backward step, are computed by

$$\boldsymbol{\mathcal{P}}_k^{i+} = \boldsymbol{\mathcal{P}}_k^i + \boldsymbol{\mathcal{K}}(\boldsymbol{\mathcal{Z}}_\zeta - z_\zeta^i). \tag{6.15}$$

The Kalman gain $\boldsymbol{\mathcal{K}}$ and cross variance $\boldsymbol{P}_{kz}$ are given by

$$\boldsymbol{\mathcal{K}} = \boldsymbol{P}_{kz} \boldsymbol{P}_z^{-1}, \tag{6.16}$$

$$\boldsymbol{P}_{kz} = \sum_{i=0}^{2L} w_c^i (\boldsymbol{\mathcal{P}}_k^i - \bar{\boldsymbol{\mathcal{P}}}_k)(\boldsymbol{z}_\zeta^i - \bar{\boldsymbol{z}}_\zeta)^\top. \tag{6.17}$$

The mean vector and covariance matrix of the distribution $p(\boldsymbol{\mathcal{P}}_k \mid \boldsymbol{\mathcal{Z}}_0, \boldsymbol{\mathcal{Z}}_\zeta)$ are finally estimated as

$$\begin{aligned} \bar{\boldsymbol{\mathcal{P}}}_k^+ &= \sum_{i=0}^{2L} w_m^i \boldsymbol{\mathcal{P}}_k^{i+} \\ &= \bar{\boldsymbol{\mathcal{P}}}_k^- + \boldsymbol{\mathcal{K}}(\boldsymbol{\mathcal{Z}}_\zeta - \bar{\boldsymbol{z}}_\zeta), \end{aligned} \tag{6.18}$$

$$\boldsymbol{P}_k^+ = \boldsymbol{P}_k^- - \boldsymbol{\mathcal{K}} \boldsymbol{P}_z^{-1} \boldsymbol{\mathcal{K}}^\top. \tag{6.19}$$

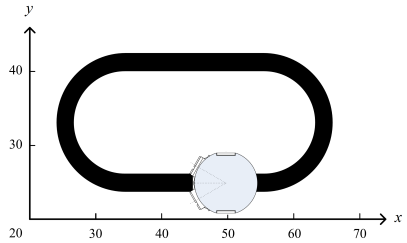
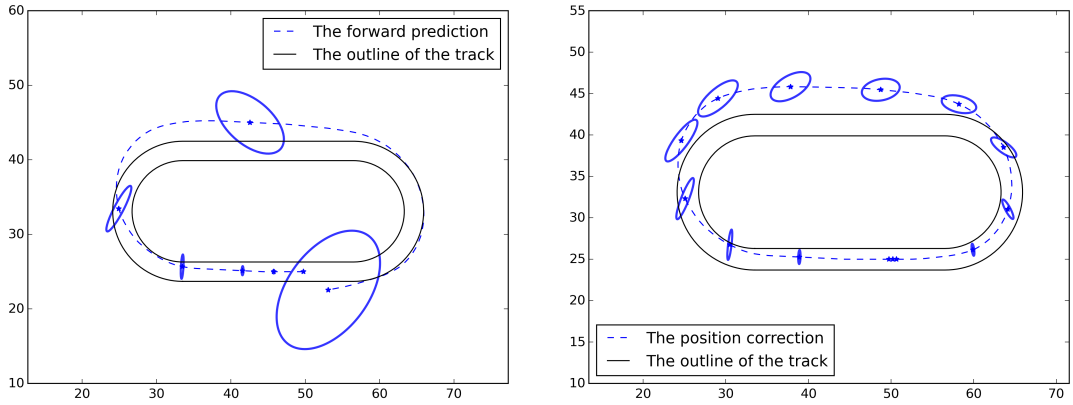


Figure 6.9: Coordinates of the track



(a) Position prediction (b) Position correction

Figure 6.10: Position prediction and correction. The blue points are means of position distributions and the ellipses represent the position uncertainty of the robot (level curve of the distribution at three standard deviations).

As a summary, the algorithm starts by predicting the poses $\boldsymbol{\mathcal{P}}_k$ until a measurement is obtained. At that point, a prediction of the measurement is done based on the computed poses $\boldsymbol{\mathcal{P}}_k$. Finally, combining both using the Kalman filter given in equations (6.18) and (6.19) yields the corrected pose for any time k along the trajectory.

6.2.4 Experimental results

The position of the mark is set to be at coordinates (45,25), and the coordinates of the track are shown in Figure 6.9. The two observations $\boldsymbol{\mathcal{Z}}_0$ and $\boldsymbol{\mathcal{Z}}_\zeta$ are (45, 25, -π). The covariance matrices $\boldsymbol{\mathcal{R}}$ and $\boldsymbol{\mathcal{Q}}$ are set to

$$\boldsymbol{\mathcal{R}} = \begin{bmatrix} 10^{-5} & 0 & 0 \\ 0 & 10^{-5} & 0 \\ 0 & 0 & 5 \times 10^{-5} \end{bmatrix}, \qquad \boldsymbol{\mathcal{Q}} = 8 \times 10^{-6} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The other three parameters are set to $\alpha = 0.00001$, $\beta = 2$, $\kappa = 0$.

The predicted position of the robot before reaching the mark is shown in Figure 6.10(a). As time goes, the uncertainty increases. After pose smoothing, the position correction is shown in Figure 6.10(b). The estimated positions on the top are out of the outer line of the track with large variances, but the estimated positions near the initial and final positions are more accurate, as expected.

6.3 Mapping incorporating the uncertainty of robot poses

The mapping methods based on MRF and GRF proposed in Chapters 4 and 5 compute posterior distributions from a prior term and a likelihood term assuming that the precise position of the robot is known. When the uncertainty of robot poses is considered, the problem is how to incorporate that uncertainty into the two terms. In the MRF-based methods, the map is divided into grid cells, and uses a global coordinate system independent from the robot. The prior distribution is defined from the local neighbourhood of the grid cells. In the GRF-based methods, the prior distributions depend on the relative positions between different points. However, the chosen points in observed space are the central points of observed grid cells. The points in unobserved space are chosen arbitrarily and do not depend on robot poses. Without pose uncertainty, the likelihood term is derived directly from the measurement model $p(z \mid m)$. When the robot pose is uncertain, the measurement model at the time step k is $p(z^k \mid m, \boldsymbol{\mathcal{P}}_k)$. Based on the law of total probability, the uncertainty of the robot pose can be incorporated in the sensor uncertainty by

$$p(z^k \mid m) = \int p(z^k \mid m, \boldsymbol{\mathcal{P}}_k) p(\boldsymbol{\mathcal{P}}_k) \mathrm{d}\boldsymbol{\mathcal{P}}_k. \tag{6.20}$$

In this thesis, the map is divided into grid cells, but it is not easy to integrate the probability in each grid cell. This problem can be solved by sampling from the distribution $p(\boldsymbol{\mathcal{P}}_k \mid \boldsymbol{\mathcal{Z}}_0, \boldsymbol{\mathcal{Z}}_\zeta)$ and averaging over all the pose samples. Since every pose sample has no uncertainty, the measurement model without pose uncertainty can be used directly. Taking the advantage of sigma points $\boldsymbol{\mathcal{P}}_k^{i+}$ obtained in the previous section, equation (6.20) can be rewritten as

$$p(z^k \mid m) = \sum_{i=0}^{2L} w_m^i p(z^k \mid m, \boldsymbol{\mathcal{P}}_k^{i+}). \tag{6.21}$$

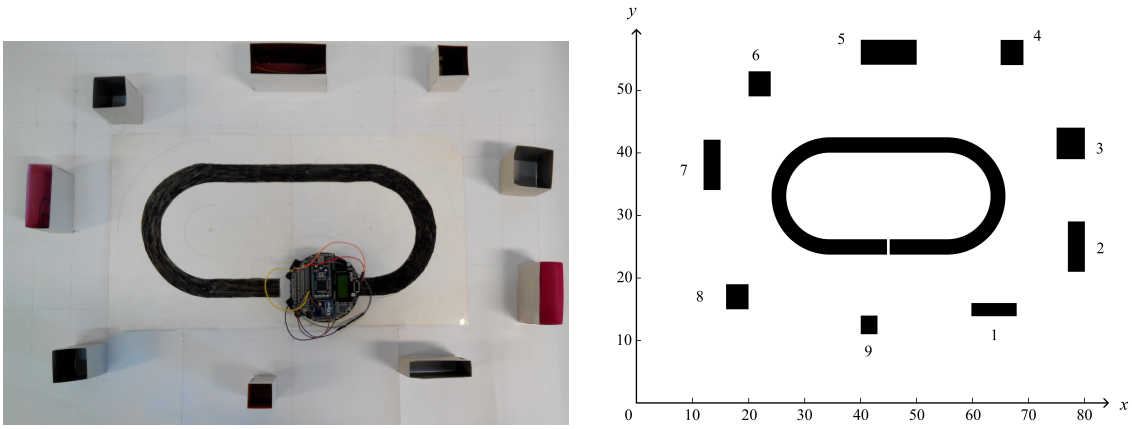
Assuming m denotes a whole grid map and the state of one grid cell is denoted by m_i, the probability $p(z^k \mid m_i)$ can be obtained by

$$\begin{aligned} p(z^k \mid m_i) &= \sum_{m \setminus m_i} p(z^k \mid m) p(m \setminus m_i \mid m_i) \\ &= \sum_{m \setminus m_i} \sum_{i=0}^{2L} w_m^i p(z^k \mid m, \boldsymbol{\mathcal{P}}_k^{i+}) p(m \setminus m_i \mid m_i) \\ &= \sum_{i=0}^{2L} \sum_{m \setminus m_i} w_m^i p(z^k \mid m, \boldsymbol{\mathcal{P}}_k^{i+}) p(m \setminus m_i \mid m_i) \\ &= \sum_{i=0}^{2L} w_m^i p(z^k \mid m_i, \boldsymbol{\mathcal{P}}_k^{i+}), \end{aligned} \tag{6.22}$$

where $m \setminus m_i$ means the whole map m without m_i. After the uncertainty of robot poses are incorporated, the proposed methods in the previous chapters can be implemented as usual.

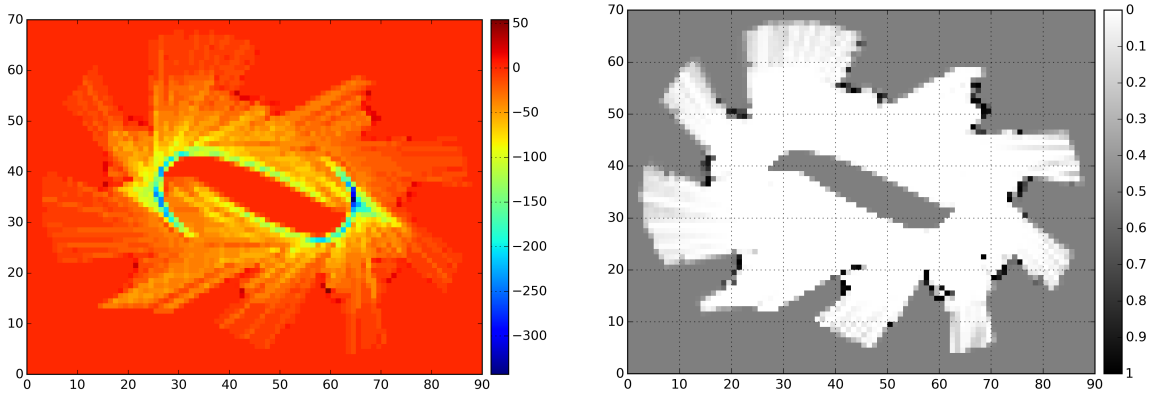
6.4 Mapping in the static environment

The static map shown in Figure 6.11(a) is built to illustrate the proposed methods for static environments, where there are some static objects with different shapes and sizes. The coordinates of the map are shown in Figure 6.11(b). Two IR sensors on the robot can measure distances to the objects. The prior occupancy



(a) Static map (b) Coordinates of the map

Figure 6.11: Static map and its coordinates

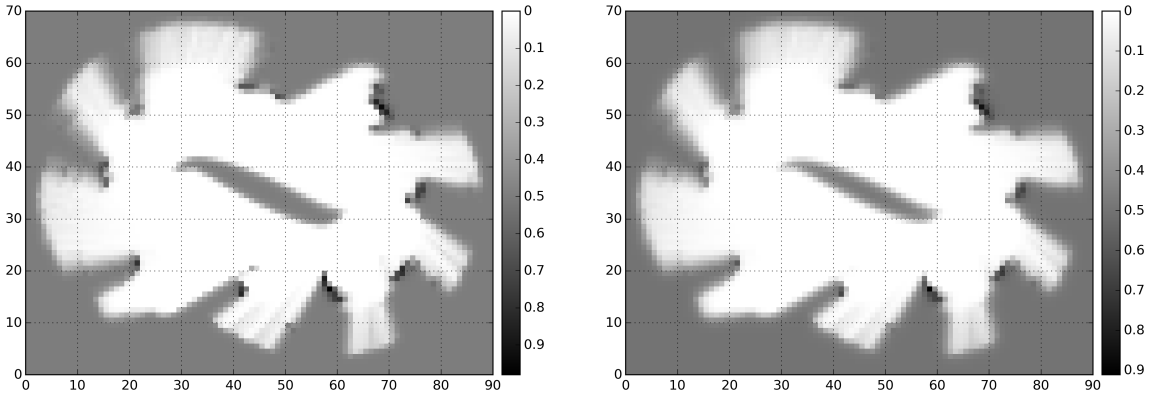


(a) Log odds occupancy observation (b) Occupancy probabilities

Figure 6.12: Log odds occupancy observation and occupancy observation of the static experimental environment. (a): The colour indicates the occupancy probability in log odds form. (b): The darkness indicates the occupancy probability.

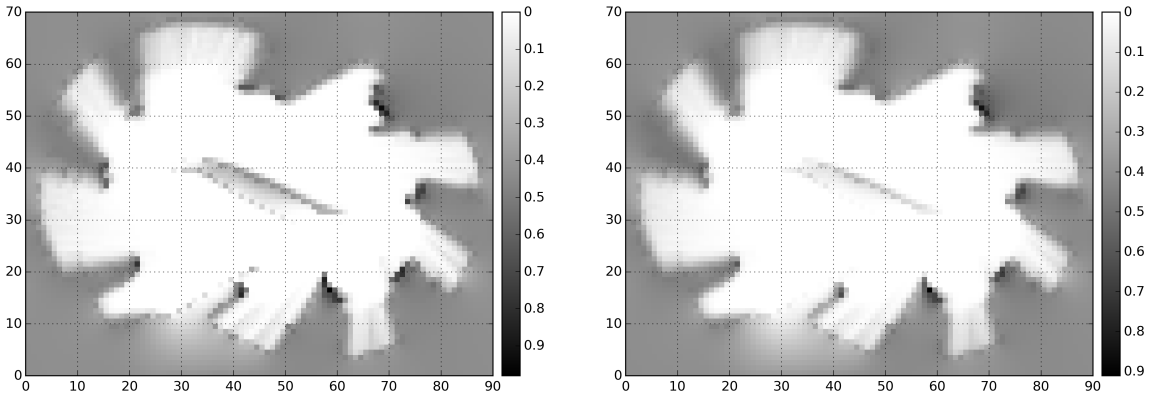
probability is set to 0.5. Regarding the sensor model, the occupancy and free probabilities of the occupied grid cells in the measurement range are set to 0.998 and 0.002, respectively. The two probabilities of the free grid cells in the measurement range are set to 0.008 and 0.992. The two probabilities for grid cells outside the measurement range are set to 0.5.

Figure 6.12(a) shows the log odds occupancy observations O_i of grid cells after the robot finishes one loop incorporating the pose uncertainty. Grid cells with values of 0 are not observed. Grid cell with values larger than 0 are observed occupied, and grid cells with value less than 0 are observed free. In order to see them clearly, the corresponding occupancy probabilities obtained by applying the logistic function are shown in Figure 6.12(b), which are the results of the classical occupancy grid mapping. In this figure, the observed states of the grid cells are clear, and there is a large uncertainty on the border of the observed space.



(a) $\sigma^2 = 0.01$ (b) $\sigma^2 = 0.03$

Figure 6.13: Results of the MRF-based filter with different σ^2 for the static experimental environment



(a) $\sigma^2 = 0.01$ (b) $\sigma^2 = 0.03$

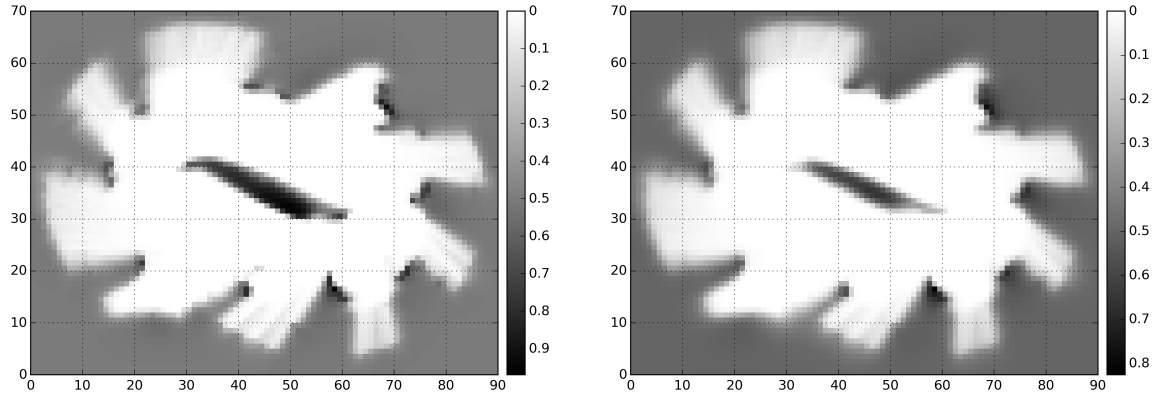
Figure 6.14: Results of the MRF-based prediction with different σ^2 for the static experimental environment

6.4.1 Filtering based on MRF

The results of the MRF-based filter with different variances σ^2 of the log odds O_i are shown in Figure 6.13. When the variance is small, there is no much difference from Figure 6.12(b). As the variance increases, there is more and there is a larger influence from neighbouring cells. As a consequence, more occupied grid cells disappear.

6.4.2 Prediction based on MRF

Figure 6.14 shows the results of the MRF-based prediction of unobserved space with different variances σ^2. With a larger value of σ^2, the result is smoother. With the same variance σ^2, the results of observed space are similar to those shown in Figure 6.13. However, the observed space tends to spread further. The unobserved space behind the occupied space is influenced by the objects and predicted occupied. Meanwhile, the unobserved space in the middle is predicted free but with large uncertainty.



(a) $\sigma^2 = 0.5$ (b) $\sigma^2 = 1$

Figure 6.15: Results of the GRF-based prediction with different σ^2 for the static experimental environment

6.4.3 Prediction based on GRF

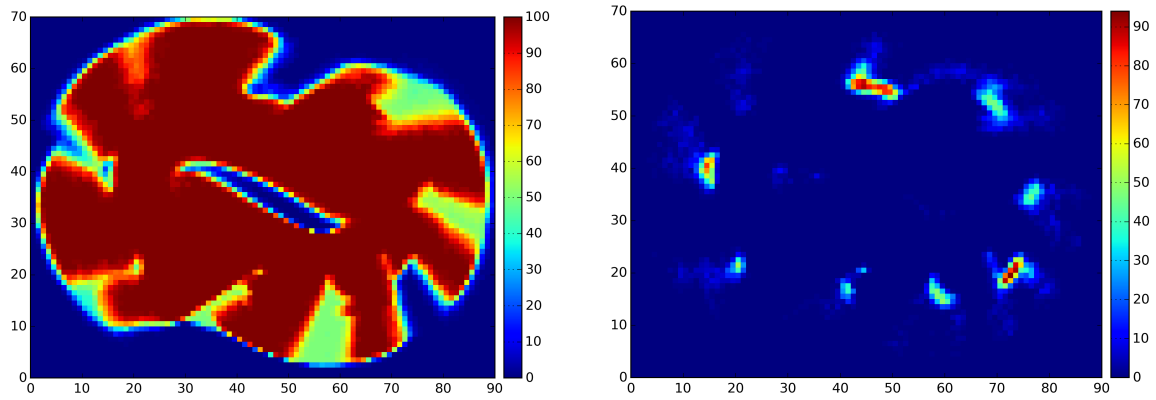
For the GRF-based prediction, the Ornstein–Uhlenbeck covariance function was selected with length-scale ℓ and signal variance σ_f^2 are set to 1, and the means are set to 0. The results with different variances σ^2 are shown in Figure 6.15, where the observed space is also smooth. Most of the unobserved space outside is essentially just the the prior. Regarding the unobserved space in the center, only the border is predicted free. Most of this area, which was expected to be predicted free, is actually predicted occupied.

6.5 Mapping in the dynamic environment

For the dynamic environment, the objects with labels 1, 3, 4, 6, 8, and 9 in Figure 6.11(b) appear and disappear from their positions with different frequencies. The object 1 changes its state at every loop and the subsequent dynamic objects change their states every 2, 5, 10, 20 and 50 loops, respectively. The objects 2, 5 and 7 are still static. The robot follows the track for 100 loops, and an occupancy grid map is built for every loop. Grid cells with posterior occupancy probabilities larger than 0.5 are assumed to be observed occupied. While posterior occupancy probabilities less than 0.5 are assumed to be observed free. The total times the grid cells are observed free or occupied are shown in Figure 6.16. Most of the space in the middle is observed free many times. The space behind the dynamic objects has about half a chance to be observed, and the space behind the static objects is never observed. For the static objects, their borders are observed partially. Because of the uncertainty of sensors and robot poses, the space around the objects is sometimes observed occupied. Similarly to most of the observed space, the number of observed times for dynamic object 6 in Figure 6.16(b) is close to 0, possibly due to the small size of the object combined with the fact that the robot is turning when the range finder crosses the object.

6.5.1 Parameter estimate without prior knowledge

The parameter estimation without prior knowledge is shown in Figure 6.17 where each observed grid cell is estimated individually. Figure 6.17(a) and Figure 6.17(b) represent the transition probabilities of the Markov chain of each pointing space. The unobserved space is covered by asterisks and has no estimates of transition probabilities. For free space, the occupied-to-occupied probability (a_{11}) is close to zero while the free-to-free a_{22} is close to one, as expected. The static objects 2, 5 and 7 have high occupied-to-occupied



(a) Observed free (b) Observed occupied

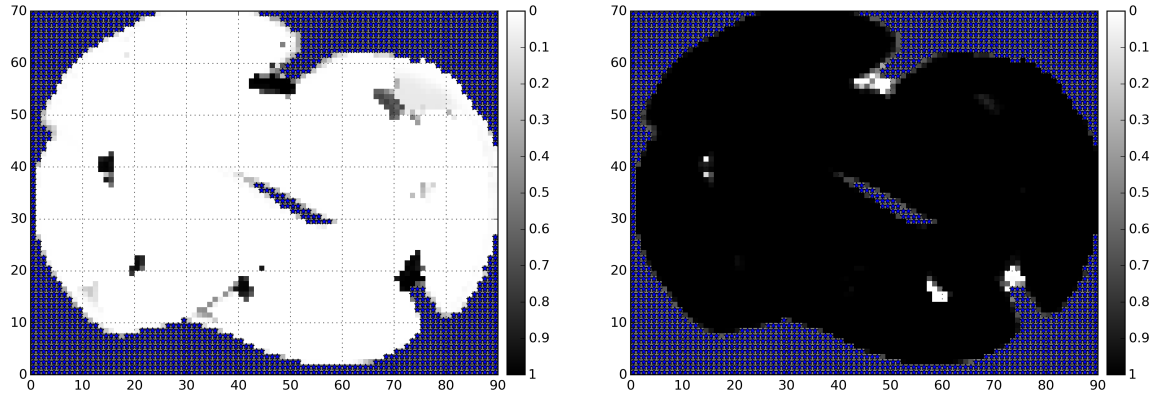
Figure 6.16: Total observed times for the dynamic experimental environment. (a): The colour indicates the number of times one grid cell is observed free. (b): The colour indicates the number of times one grid cell is observed occupied.

probability and low free-to-free. This way, Markov chain tends quickly to the occupied state which makes sense for static objects. The dynamic object 1 has the opposite behaviour, with low occupied-to-occupied and high free-to-free probabilities. This means that the state alternates between free and occupied quickly. For the dynamic objects 3 and 4, the colour becomes darker corresponding to slower dynamics. Because of the lack of observations, the dynamic object 6 is estimated as free space. The other two dynamic objects 8 and 9 change their states slowly. The space behind these dynamic objects has fewer observations, the corresponding areas are darker than the space with more free observations. The space behind static objects 2 and 5 is never observed, and therefore there is no estimate. Because of the uncertainty of the robot pose, the space behind the static object 7 has a similar estimate to free space.

Given the transition probabilities obtained, the overall expected duration of free and occupied states is shown in Figure 6.18 in linear and logarithm scales. The free space has a long overall expected duration as expected. Because of the sensor noise and uncertainty of the robot pose, the static objects have long expected durations but are surrounded by shorter ones. The dynamic objects with higher switching frequencies have shorter overall expected durations. Figure 6.18(b) shows the overall expected duration in log scale, where the duration difference between dynamic objects is more clear. The border of the observed space has a low expected duration due to sensor noise and uncertainty in the robot pose.

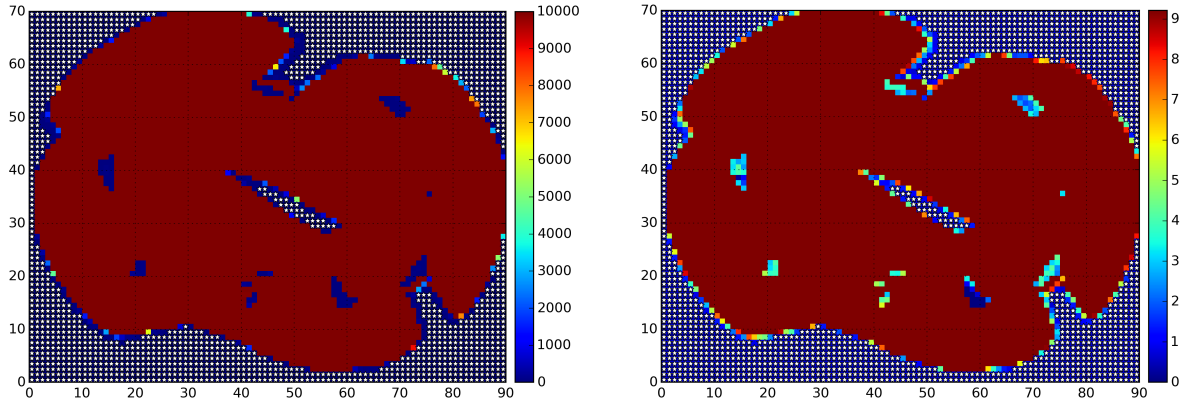
6.5.2 Prediction based on MRF

For the MRF-based prediction, the initial values of the parameters $\boldsymbol{a}_1, \boldsymbol{a}_2$, $\boldsymbol{\rho}_1$ are set to 0.5, and the temperature parameter $\mathcal{T}$ is set to 20. The maximum number of iterations of the optimization process is set to 300, and the results are shown in Figure 6.19. For most of the space, all the observations are free. As discussed in Section 5.1.5, the corresponding parameter a_{22} converges very quickly and a_{11} has less chance to be optimized. As a result, the parameter a_{11} for most of the free space in Figure 6.19a is close to the initial value 0.5. In order to decrease the convergence speed of a_{22}, for the space with more than 95 free observations, 45 of them are replaced by 45 fake observations corresponding to unknown space. The new estimation is shown in Figure 6.20, where the estimate of a_{11} of the observed free space becomes better. As to the objects, the estimates are similar to those obtained without prior knowledge. The use of a prior in the MRF has produced a smooth estimation. In Figure 6.20(b), most of the observed free space is



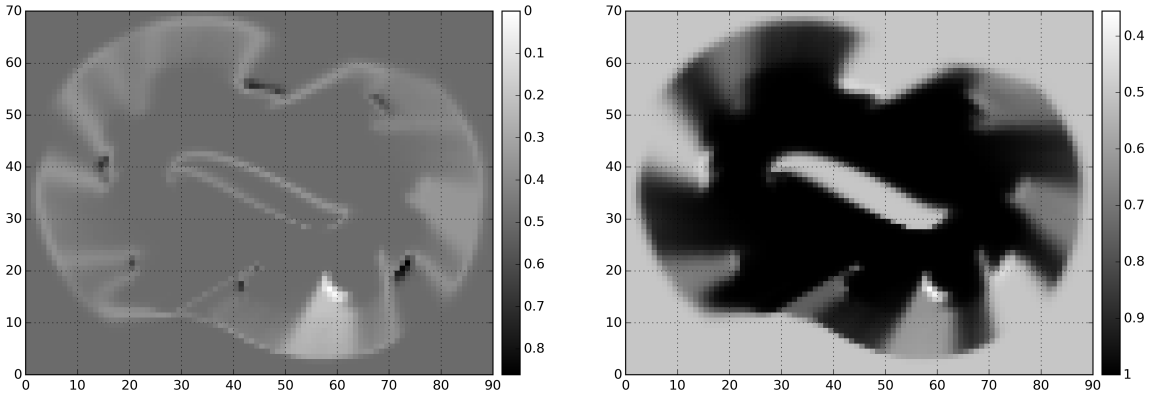
(a) a_{11}: staying in occupied state (b) a_{22}: staying in free state

Figure 6.17: HMM parameter estimation without prior for the dynamic experimental environment. (a): The darkness indicates the probability of staying in occupied state. (b): The darkness indicates the probability of staying in free state.



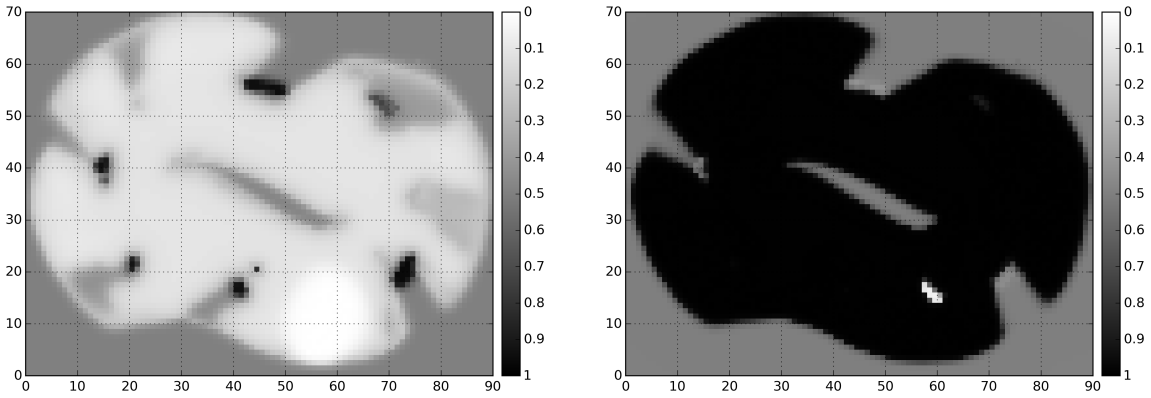
(a) Overall expected duration (b) Overall expected duration in log scale

Figure 6.18: Overall expected duration without prior for the dynamic experimental environment. (a): The colour indicates the overall expected duration. (b): The colour indicates the overall expected duration in log scale.



(a) a_{11}: staying in occupied state (b) a_{22}: staying in free state

Figure 6.19: HMM parameter estimation of the MRF-based prediction for the dynamic experimental environment

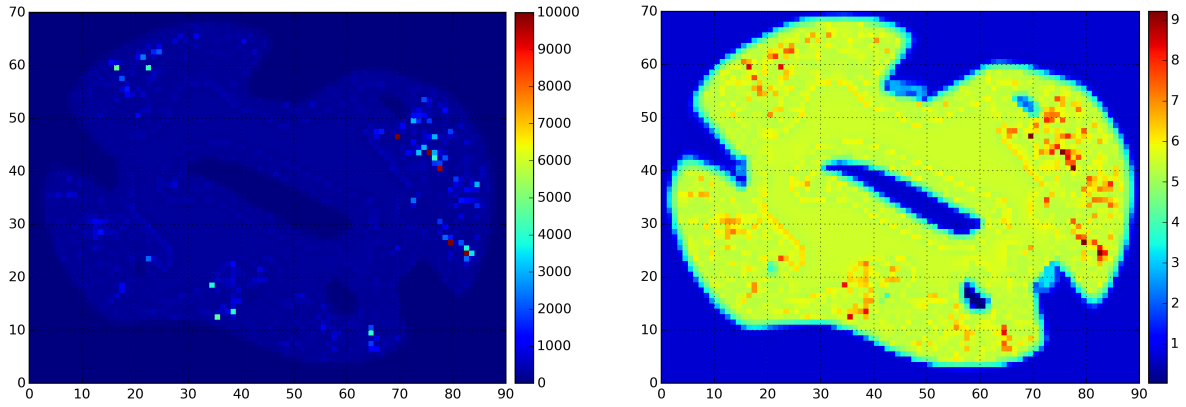


(a) a_{11}: staying in occupied state (b) a_{22}: staying in free state

Figure 6.20: HMM parameter estimation of the MRF-based prediction for the dynamic experimental environment

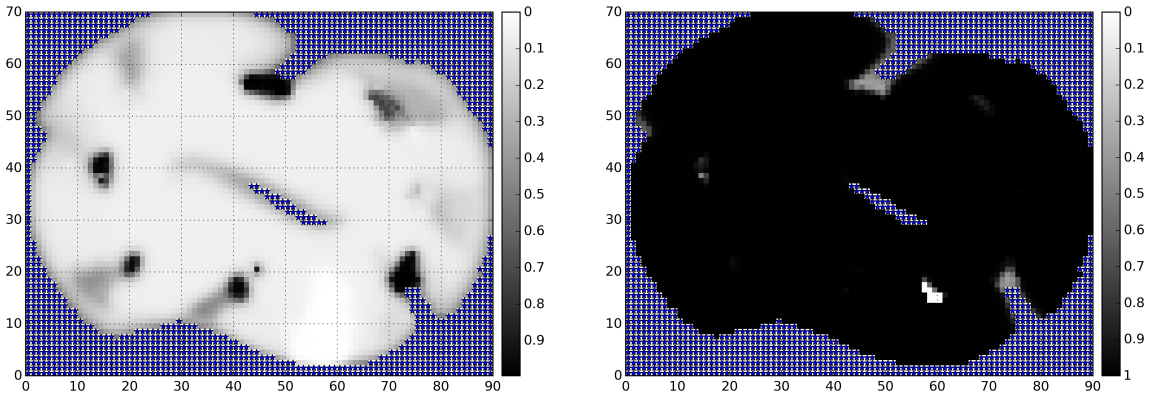
estimated correctly. The objects 1 and 4 with higher frequencies are correctly estimated. The objects with slower dynamics and the static objects have similar characteristics, tending to remain in the same state with high probability, but show differences regarding the space hidden by the objects. The static objects alway hide this space, which is then explained by the prior, while the slow dynamic objects only hide the space behind them when they are present. As a consequence, the hidden space is basically the prior in the parameter a_{11}, but similar to free space in the parameter a_{22}.

The overall expected duration is shown in Figure 6.21(a), where the difference between objects and free space is not clear. The expected duration in log scale is shown in Figure 6.18(b). The dynamic objects 3 and 6 are mostly invisible and have similar overall expected durations to the free space. The dynamic objects with higher switching frequencies, 1 and 4, have shorter overall expected durations and are clearly visible in log scale. Because the transition probabilities of the unobserved space are 0.5, the corresponding overall expected duration is short.



(a) Overall expected duration (b) Overall expected duration in log scale

Figure 6.21: Overall expected duration estimation of the MRF-based prediction for the dynamic experimental environment



(a) a_{11}: staying in occupied state (b) a_{22}: staying in free state

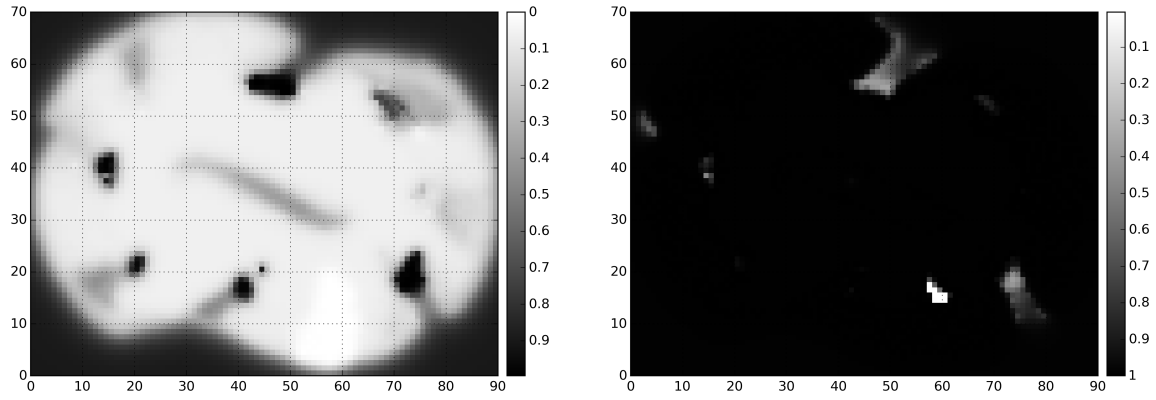
Figure 6.22: HMM parameter estimation of the GRF-based prediction for observed space in the dynamic experimental environment

6.5.3 Prediction based on GRF

As to the GRF-based prediction, the initial values of the probabilities are set to 0.5, the length-scale ℓ of the covariance function is set to 3 and the signal variance σ_f^2 is set to 25. The mean vectors $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_1^*$ of the occupied-to-occupied probabilities for the training and test points are set to $\log 9$, which corresponds to a probability of 0.9. The mean vectors $\boldsymbol{\mu}_2$ and $\boldsymbol{\mu}_2^*$ are set to $\log 99$, which corresponds to a probability of 0.99. Similarly to what was done in the MRF approach, the space with more than 95 free observations are also replaced by fake observations corresponding to unknown space. The maximum number of iterations of the optimization process is 1500. As shown in Figure 6.22, the estimate of observed space is very similar to the one obtained in Figure 6.20. The main difference is that the static objects in Figure 6.22(b) are more clearly visible than those obtained in Figure 6.20(b).

The prediction for unobserved space is shown in Figure 6.23. In Figure 6.23(a), the border of the observed space is a little fuzzy and the estimate of most of the unobserved space is similar to the prior. In Figure 6.23(b), most of the parameters on the borders of observed space are close to 1, and the parameters of

the unobserved space near these areas are also predicted to be close to one. Since the parameters of the borders of observed space near the static objects are close to 0.5, the darkness near these areas is lighter. Similarly to Figure 6.23(a), the prediction of the other unobserved space is similar to the prior.

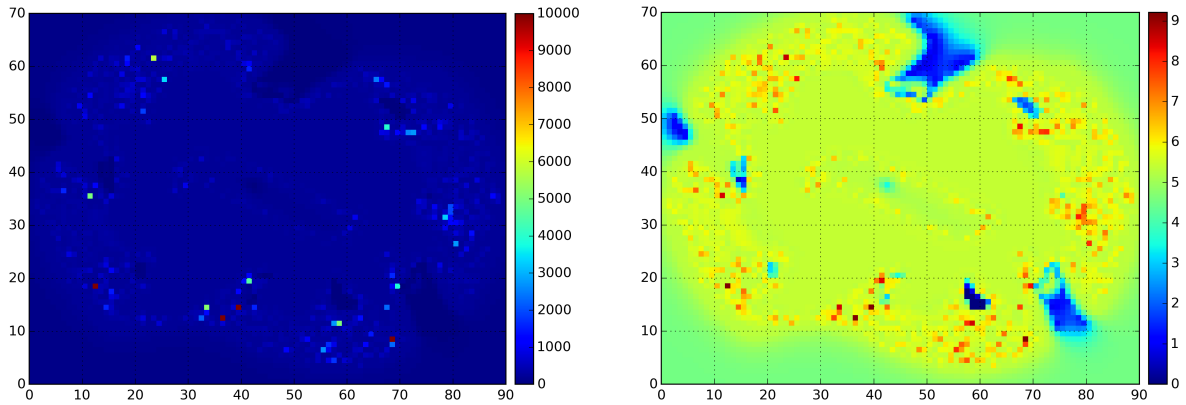


(a) a_{11}: staying in occupied state

(b) a_{22}: staying in free state

Figure 6.23: HMM parameter prediction of the GRF-based prediction for unobserved space in the dynamic experimental environment

The overall expected duration is shown in Figure 6.24(a) and the version in log scale is shown in Figure 6.24(b). For the observed space, Figure 6.24(b) is similar to Figure 6.18(b). The observed free space, the static objects, and the low dynamic objects have long overall expected expectations. As was the case in the MRF approach, the overall expected expectations of the dynamic objects 3 and 6 are very long. For the unobserved space behind the static objects, the overall expected duration is short. The rest of the unobserved space has a long overall expected duration.



(a) Overall expected duration

(b) Overall expected duration in log scale

Figure 6.24: Overall expected duration of GRF-based prediction for the dynamic experimental environment

6.6 Comparison

When considering static environments, the MRF-based filter can smooth the observed space, and its main advantage is that it can be computed in few seconds in our experiment setup. The MRF-based prediction

Table 6.1: Comparison between results for the static experimental environment. TF= True free, TO= True occupied, FF= False free, FO= False occupied, UN=Unknown.

Classification	TF	TO	FO	FF	UN
Classical OGM (Figure 6.12(b))	2950	26	79	109	3136
MRF-based filter (Figure 6.13(a))	3241	27	77	120	2835
MRF-based prediction (Figure 6.14(a))	5954	30	101	141	74
GRF-based prediction (Figure 6.15(a))	3578	45	128	119	2430

Table 6.2: Comparison between results for the dynamic experimental environment. TF = True free, TO = True occupied, FF = False free, FO = False occupied, TLD = True low dynamic, FLD = False low dynamic, THD = True high dynamic, FHD = False high dynamic, UN = Unknown.

Classification	TF	FF	TO	FO	TLD	FLD	THD	FHD	UN
Independent assumption (Figure 6.17)	4362	145	18	13	0	35	4	152	1571
MRF-based prediction (Figure 6.20)	3902	146	13	9	0	15	2	687	1526
GRF-based prediction (Figure 6.23)	4648	146	16	11	0	752	2	725	0

can deal with unobserved space, but has the disadvantage that it takes longer. The GRF-based prediction can predict the unobserved space and be implemented online. Even though it takes longer time than the filter, the time is sufficient for the map to be done in real time in our experiment setup.

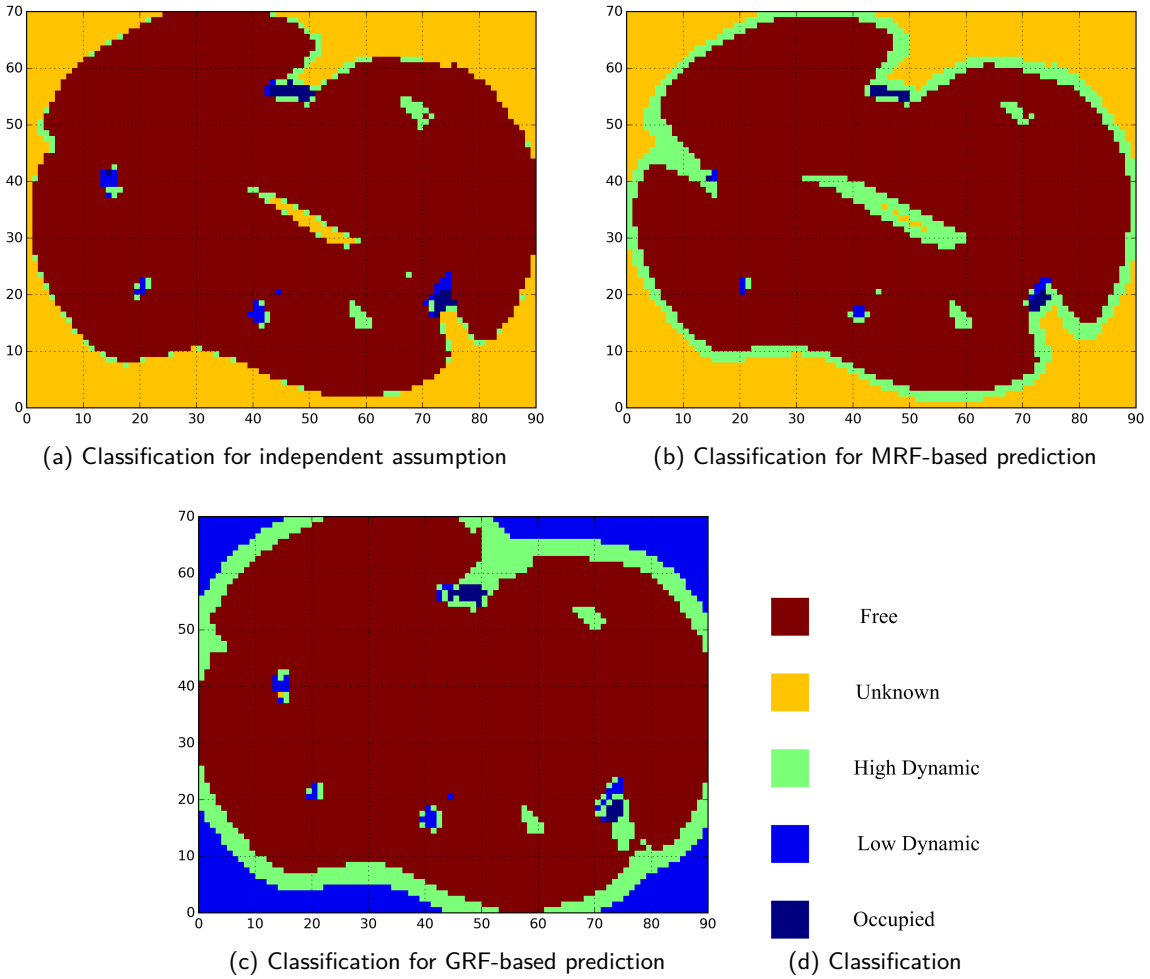
When the occupancy probability in the results for static experimental environments is between 0.47 and 0.53, more than 0.53, or less than 0.47, it is classified as unknown, occupied or free. The comparison between Figures 6.12(b), 6.13(a), 6.14(a) and 6.15(a) is shown as Table 4.1. These figures represent the classical occupancy grid mapping (OGM) and the proposed methods. All the proposed methods can predict unknown space and the MRF-based method predicts more space. The other two proposed methods produce similar results. Because the central unknown space is predicted occupied for GRF-based method, the corresponding false occupied (FO) is high.

In dynamic environments, the parameter estimates of the HMMs without prior knowledge obtained in Section 6.5.1 are a little noisy. The MRF-based prediction of Section 6.5.2 produces similar estimates for the objects to the previous method. Because of the prior and the lack of free observations for the occupied space, the parameter a_{22} for the occupied space cannot be estimated correctly. For the observed space, the results of the three methods give similar results. The estimate of most of unobserved space is similar to the prior knowledge in the GRF-based prediction.

Based on the classification as Table 5.4, the objects 1, 3, 4 and 6 in the experimental map are high dynamic, and the objects 8 and 9 are low dynamic. The classification results for Figures 6.17, 6.20 and 6.23 are shown as Figure 6.25 and Table 6.2. Because of pose uncertainty, all the true low dynamic (TLD) space is wrong. As to the MRF-based prediction, the free space on the border is estimated as high dynamic. As a result, it has less true free (TF) space and more false high dynamic (FHD) space. The GRF-based method can predict more free space correctly. Because of the prior low dynamic assumption, there are more false low dynamic (FLD) space. The proposed methods can smooth the map, there must be more FHD space.

6.7 Summary

The 3pi robot is the base of the experimental platform used to test the proposed methods. Two IR sensors are used to obtain data from the environment. The data is sent to the PC through two Xbee modules



(a) Classification for independent assumption

(b) Classification for MRF-based prediction

(c) Classification for GRF-based prediction

(d) Classification

Figure 6.25: Classification of the results for the dynamic experimental environment

and the PC runs the algorithms to build the maps. The proposed methods in Chapters 4 and 5 assume known poses for the robot. However, the robot pose is not known in the experimental environments. The localization problem is simplified by assumming that the robot follows a track continuously and detects a mark every loop, which allows its precise location to be known at that point. As the robot follows the track, the speeds and measurement ranges are sent to the PC. When the robot meets the mark twice, the pose can be corrected using the measurements and a trajectory closure constraint. The uncertainty of the robot poses is incorporated into the uncertainty of the IR sensors, and the proposed methods can be implemented as usual as if the robot location were known. First, the methods for static environments are tested. Several static objects with different shapes and sizes spread around the track. The MRF-based filter can produce a smooth map for the observed space. The MRF-based and GRF-based predictions can predict unobserved space. In the dynamic environment, some of the static objects become dynamic with different switching frequencies. Markov chains are used to model the transition probabilities vary from point to point in the map, but some correlation depending on the distance is assumed. In the MRF-based methods this correlation is introduced by considering the immediate neighborhood of every grid cell, while in the GRF-based methods a covariance function is used for that purpose. The GRF-based and MRF-based predictions produce smoother results than the estimation without prior knowledge. The experimental results presented in this chapter demonstrate viability of the the proposed methods for static and dynamic environments.

In a real problem, the size of the map will increase with time and there are more parameters to be estimated.

For the MRF-based filter in static environments, its computational complexity increases linearly with the size of built maps. Since this method can build local maps, its computational complexity does not increase too much. The GRF-based prediction for static environments can also do local mapping. Its computational complexity scales quadratically with the number of training and linearly with the number of test points in local maps. However, the MRF-based prediction for static environments has quadratic computational complexity with the size of the whole map. The MRF-based prediction for dynamic environments has the same computational complexity as the static environment. As to the GRF-based prediction for dynamic environments, its computational complexity is the same as the GRF-based method for static environments.

7 Conclusions

7.1 Summary

In this thesis, the mapping problem is approached considering a static as well as a dynamic environment. The proposed techniques follow a Bayesian approach using Markov random fields (MRFs) and Gaussian random fields (GRFs), and were validated in simulation and experimental results. The experimental results show that the proposed methods can be implemented in practical situations.

In Chapter 4, occupancy probabilities in log odds form are used to represent static maps and regarded as random variables. The observations of random variables are the result of the classical occupancy grid mapping. The first approach is an MRF-based filter where a map is divided into grid cells and unobserved grid cells are also considered in the likelihood function. Maximizing the posterior distribution of an MRF model requires a matrix inversion. The nonzero elements in this inverse matrix do not change with the increasing map size. When all the variances of observations are assumed to be the same, these nonzero elements can be obtained by solving another inverse matrix with a small size. The second approach is an MRF-based prediction where the variances of observations of unobserved grid cells are set to infinity. As a result, the unobserved grid cells do not take any effect on the likelihood and can only learn from observed space. The solution can also be obtained by solving an inverse matrix. With different grid cells observed,

the inverse matrix is different. However, the inverse matrix only needs to be computed at the beginning, and the following inverse matrices can be computed recursively based on the Sherman-–Morrison equation when new grid cells are observed. Finally, an alternative prediction approach based on GRF is proposed, which can work in continuous space. The solution can be obtained by the well-known prediction equation with an inverse matrix problem in the Gaussian processes with noisy observations. Another recursive algorithm is also proposed instead to avoid the inverse matrix problem. Meanwhile, this method can build maps with high local resolution. The smoothness can be ensured by overlapping enough observations, which depends on the covariance function. The simulation results for the three methods are presented in the corresponding sections.

In Chapter 5, Markov chains are applied to model dynamic environments. The transition matrix of a Markov chain can be represented by two parameters. These two parameters are assumed to be mutual independent but are assumed to be dependent of the parameters in their neighbourhood. First, an MRF-based prediction is proposed, which can estimate the parameters for observed and unobserved space. The EM algorithm is applied to solve this problem and a line search method is used to search for the maximum in the maximizing step of the EM algorithm. Finally, a GRF-based prediction is proposed, where the parameter estimation is divided into two steps: parameter estimation for training points and parameter prediction for test points. The first step is to estimate the parameters underlying noisy observations, which is similar to the MRF-based prediction. Based on the parameters of training points, the parameters of test points can be estimated by the prediction equation in the Gaussian processes with noise-free observations. The two proposed methods are tested in simulations in the corresponding sections.

In Chapter 6, an experimental platform is built to validate the proposed methods for static and dynamic environments. A 3pi robot equipped with two IR sensors is used to obtain data from experimental environments. The data is sent to a PC by two Xbee modules. In real environments, the uncertainty of robot poses is incorporated in sensor uncertainty, and the proposed methods with known robot poses can be implemented as usual. In order to estimate the uncertainty of robot poses, a track with a mark is designed, and the unscented Kalman filter is applied to smooth trajectories with a loop closure constraint. In the static experimental environment, some static objects are around the track, and the robot follows the track one loop. In the dynamic experimental environment, some objects become dynamic with different changing frequencies, and the robot follows the track multiple loops. Finally, the experimental results for different methods are presented and discussed.

7.2 Future work

In the future, some research questions can be solved to refine the proposed methods and they are listed as follows:

- **Local mapping of the MRF-based prediction for static environments**. In Section 4.3, the MRF-based filter can be implemented locally. When one grid cell needs to be updated, some other grid cells near it should be considered. When all the grid cells are observed, the MRF-based prediction in Section 4.4 becomes an online filter. It means the MRF-based prediction could be done locally in the area with more grid cells observed. Similarly to the GRF-based prediction, the smooth of maps can also be ensured by overlapping data. The size of the overlapped area depends on the variance and the number of observed grid cells. When the variance is small, one grid cell has low uncertainty, and fewer grid cells are correlated. As a result, the size of the overlapped area should be small. When there are unobserved grid cells near it, the dependence on the unobserved grid cells will spread to other observed grid cells, and the overlapped area will enlarge.

- **Active sampling of the GRF-based prediction for static environments**. In Section 4.5, the GRF-based prediction works in continuous space. However, observed space is divided into grid cells to choose training points, and the discretization requires much memory. This method is more complex computationally with more training data. In [OR12], only one occupied point and one free point are chosen for every laser beam in continuous space. In [WVW12], active sampling is proposed to reduce the number of training points. These sampling strategies may arise another problem that the coordinates of training points depend on robot poses. In this case, the uncertainty of robot poses should be considered in prior distributions [OR12].

- **Alternative to the Ornstein–Uhlenbeck kernel function of the GRF-based prediction for static environments**. In Section 4.5, the Ornstein–Uhlenbeck kernel function is applied. Figure 4.14 shows that Ornstein–Uhlenbeck kernel function is better than the other two. In experimental results shown in Figure 6.15, the space around the observed space in the middle is observed free. However, the observed space is predicted occupied. It means the Ornstein–Uhlenbeck kernel function does not ensure the unknown points are always predicted free if all the training points are free. In this thesis, all the parameters are chosen manually. The parameters can be learnt by maximizing marginal likelihood [RW06]. If the map size is very big, different regions may have different parameters. As a result, the parameters should be optimized locally.

- **Backward procedure in the EM algorithm could be discarded**. In the proposed methods for dynamic environments in Chapter 5, the search process is very slow, and the E step of the EM algorithm takes a long time. In the Baum–Welch algorithm, the E step includes a forward procedure and a backward procedure. Normally the backward procedure is discarded to reduce the computational complexity in online learning algorithms [MD08], [RDH+16]. Without the backward procedure, the forward procedure can be computed recursively. The disadvantage is that a lot of information is lost, and it requires more data to obtain reliable parameter estimates.

- **Scaling in a real problem**. As discussed in Section 6.7, the computational complexity for the proposed methods that can not do local mapping will increase with the map size. One possible way is to refine these methods, such as modifying them to be local mapping methods or discarding the complex step as discussed in the previous paragraphs. Another possible way is to reduce the data size. It does not change the built map so much to drop similar values at similar locations [WVW12]. The data with fewer observations in dynamic environments can be discarded because it reduces the convergence speed for the proposed methods and the corresponding parameters can not be optimized.

7.3 Concluding remarks

Taking the advantage of the usage of the log odds form and virtual observations, the MRF-based filter and GRF-based prediction proposed in this thesis for static environments have lower computational complexity than the state-of-the-art GRF-based methods. On the other hand, the proposed MRF-based prediction can not be applied to do local mapping due to its high computational complexity for large maps. Despite this limitation, all three methods can be used to smooth maps and predict unobserved space. However, the smoothing should be done with care, as small objects can be regarded as noise and be filtered out.

The methods available in the literature to tackle similar problems, in dynamic environments and using Markov chains, assume that each point in space is independent of the others. When compared with them, the methods proposed in this thesis can build smooth maps and predict the dynamic behaviour of unobserved space. However, they have higher computational complexity and can only be applied offline.

For the proposed MRF-based methods, the parameters only affect the weight between prior knowledge and

observations, and they produce reasonable maps without parameter tuning. For the GRF-based methods, however, the signal variance and length scale parameters should be carefully selected before running the algorithms.

For static environments, the observations are represented as occupancy grid maps (OGMs), while for dynamic environments the observations are sequences of OGMs. Normally, the methods for simultaneous localization and mapping (SLAM), such as Kalman filter SLAM and fast SLAM, produce OGMs. As a result, the proposed methods can be implemented together with other mapping techniques where OGMs are available.

In principle, all the proposed methods can be applied with other sensors whenever range information can be extracted, such as with cameras by matching features or pixels in multiple images.

A

Basic probabilistic concepts and common distribution

This chapter introduces the basic concepts of probability theory and distributions. For more information, see [Bil08].

A.1 Basic probabilistic concepts

Probability is a mathematical tool used to represent random experiments whose results are unpredictable. In a random experiment, the set of all possible outcomes is denoted by Ω. For example, a coin toss has $\Omega = \{head, tail\}$. In many situations, it is helpful to have results that are numeric. In those cases, a random variable $X : \Omega \to \mathbb{R}$ can be defined that assigns numerical values to the outcomes. The notation $P(X = x)$, where x is a real number, is used to mean the probability of the (set of) outcomes that are mapped to the same number x by the random variable X, i.e., $P(\{\omega \in \Omega \mid X(\omega) = x\})$. For example, in coin tossing, it is possible to define a random variable that assigns 0 to heads and 1 to tails. The function

$P(X = 0)$ means the probability of heads $P(\{heads\})$. Usually a probability function $p(x)$ is defined for convenience.

For discrete random variables, the range of $p(x)$ is $[0, 1]$ and is normalized so that

$$\sum_x p(x) = 1. \tag{A.1}$$

The expected value of a random variable is defined by the weighted mean of x

$$E(X) = \sum_x xp(x). \tag{A.2}$$

Continuous random variables X have a continuum set of values. The range of the probability density function $p(x)$ is $[0, +\infty]$ and the integral over $p(x)$ is also one

$$\int p(x)\mathrm{d}x = 1. \tag{A.3}$$

The mean is defined in a similar fashion as for discrete variables by

$$E(X) = \int xp(x)\mathrm{d}x. \tag{A.4}$$

The variance of a random variable X is defined by

$$Var(X) = E[(x - E(x))^2], \tag{A.5}$$

which measures the average squared deviation from its mean.

The joint probability function of two random variables X and Y is defined by

$$p(x, y) = P(X = x \cap Y = y), \tag{A.6}$$

which represents the probability of observing an outcome in both sets $X = x$ and $Y = y$. The covariance of two random variables X and Y is defined by

$$Cov(X, Y) = E[(x - E(X))(y - E(Y))], \tag{A.7}$$

which is the expected product of their deviations. The conditional distribution of X given Y is defined by

$$p(x \mid y) = \frac{p(x, y)}{p(y)} \tag{A.8}$$

provided $p(y) \neq 0$. The random variables X and Y are said to be independent if

$$p(x, y) = p(x)p(y). \tag{A.9}$$

A.2 Common distributions

Some common distributions used in this thesis are introduced.

A.2.1 Uniform distribution

In a uniform distribution, the probability density function is constant in a given interval, and zero everywhere else

$$p(x) = \begin{cases} \frac{1}{d_2 - d_1} & \text{if } x \in [d_1, d_2], \\ 0 & \text{otherwise.} \end{cases} \tag{A.10}$$

The mean and variance are $\frac{1}{2}(d_1 + d_2)$ and $\frac{1}{12}(d_2 - d_1)^2$, respectively.

A.2.2 Exponential distribution

In a exponential distribution, the range of x is $[0, \infty)$ and the probability density is

$$p(x) = \lambda \exp\left(-\lambda x\right), \tag{A.11}$$

where λ is a parameter. The mean and variance are $\frac{1}{\lambda}$ and $\frac{1}{\lambda^2}$, respectively.

A.2.3 Gaussian distribution

An one-dimensional Gaussian distribution is defined by the probability density function

$$p(x) = \frac{1}{\sqrt{(2\pi)\sigma^2}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right), \tag{A.12}$$

where μ is the mean and σ^2 is the variance. A random variable with a Gaussian distribution is denoted by $X \sim \mathcal{N}(\mu, \sigma^2)$.

For a n-dimensional random vector $\boldsymbol{X}$ with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{K}$, its distribution is expressed as

$$p(x) = \frac{1}{\sqrt{(2\pi)^n |K|}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{K}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right), \tag{A.13}$$

and a random vector with this distribution is denoted by $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{K})$.

B

Derivation of Q function

For any non-zero probability $p(O, \boldsymbol{A})$, equation (5.43) can be rewritten as

$$\log p(O, \boldsymbol{A}) = \log p(\mathcal{M}, O \mid \boldsymbol{A}) - \log p(\mathcal{M} \mid O, \boldsymbol{A}) + \log p(\boldsymbol{A}). \tag{B.1}$$

The expectation over possible values of the underlying map sequence $\mathcal{M}$ under the current parameter estimate $\boldsymbol{A}^{(k)}$ is formulated as

$$\begin{aligned} \log p(O, \boldsymbol{A}) &= \sum_{\mathcal{M}} p(\mathcal{M} \mid O, \boldsymbol{A}^{(k)}) \log p(\mathcal{M}, O \mid \boldsymbol{A}) + \log p(\boldsymbol{A}) \\ &\quad - \sum_{\mathcal{M}} p(\mathcal{M} \mid O, \boldsymbol{A}^{(k)}) \log p(\mathcal{M} \mid O, \boldsymbol{A}) \\ &= Q(\boldsymbol{A}, \boldsymbol{A}^{(k)}) + \mathcal{F}(\boldsymbol{A}, \boldsymbol{A}^{(k)}), \end{aligned} \tag{B.2}$$

where $\mathcal{F}(\boldsymbol{A}, \boldsymbol{A}^{(k)})$ is defined by the negated sum it is replacing. This last equation holds for any value of $\boldsymbol{A}$ including $\boldsymbol{A} = \boldsymbol{A}^{(k)}$,

$$\log p(O, \boldsymbol{A}^{(k)}) = Q(\boldsymbol{A}^{(k)}, \boldsymbol{A}^{(k)}) + \mathcal{F}(\boldsymbol{A}^{(k)}, \boldsymbol{A}^{(k)}), \tag{B.3}$$

and subtracting this last equation from the previous equation gives

$$\begin{aligned} \log p(O, \boldsymbol{A}) - \log p(O, \boldsymbol{A}^{(k)}) = Q(\boldsymbol{A}, \boldsymbol{A}^{(k)}) - Q(\boldsymbol{A}^{(k)}, \boldsymbol{A}^{(k)}) \\ + \mathcal{F}(\boldsymbol{A}, \boldsymbol{A}^{(k)}) - \mathcal{F}(\boldsymbol{A}^{(k)}, \boldsymbol{A}^{(k)}). \end{aligned} \tag{B.4}$$

Gibbs' inequality tells us that $\mathcal{F}(\boldsymbol{A}, \boldsymbol{A}^{(k)}) \geq \mathcal{F}(\boldsymbol{A}^{(k)}, \boldsymbol{A}^{(k)})$, so we can obtain

$$\log p(O, \boldsymbol{A}) - \log p(O, \boldsymbol{A}^{(k)}) \geq Q(\boldsymbol{A}, \boldsymbol{A}^{(k)}) - Q(\boldsymbol{A}^{(k)}, \boldsymbol{A}^{(k)}). \tag{B.5}$$

Choosing $\boldsymbol{A}$ to improve $Q(\boldsymbol{A}, \boldsymbol{A}^{(k)})$ beyond $Q(\boldsymbol{A}^{(k)}, \boldsymbol{A}^{(k)})$ can not cause $\log p(O, \boldsymbol{A})$ to decrease below $\log p(O, \boldsymbol{A}^{(k)})$, and the marginal likelihood of the data is non-decreasing.

C

Line search methods

A line search method is used to find a minimum of a given function $f(x)$ along a descent direction iteratively,

$$x_{k+1} = x_k + \lambda_k g_k, \tag{C.1}$$

where λ_k is the step length and g_k is the search direction. Each iteration must ensure

$$f(x_{k+1}) \leq f(x_k). \tag{C.2}$$

There are many choices of λ_k which can satisfy this condition. If λ_k is not appropriate, the minimizing process will be very slow. In order to make the process fast, λ_k should satisfy the Wolfe conditions [WN99], which consists of a sufficient decrease condition and a curvature condition. The sufficient decrease condition is

$$f(x_{k+1}) \leq f(x_k) + c_1 \lambda_k \nabla f(x_k)^\top g_k, \tag{C.3}$$

where $\nabla f(x_k)$ is the gradient of $f(x)$ and $c_1 \in (0, 1)$. The curvature condition is

$$\nabla f(x_{k+1})^\top g_k \geq c_2 \nabla f(x_k)^\top g_k, \tag{C.4}$$

where $c_2 \in (0, c_1)$. Normally, the direction g_k is set to the negative gradient $-\nabla f(x_k)$. The line search method is described as Algorithm 7, where x_0 is the initial value and $step$ is the maximum iteration. In step 3, the step length λ_k will decrease from an initial value at a rate in (0, 1) until it satisfies the Wolfe conditions.

Algorithm 7 Line search method LSM($f(x)$, x_0, $step$)

Input: $f(x)$, x_0, $step$
Output: x_k

1: $k = 0$
2: **while** $k < step$ **do**
3: search λ_k which satisfies the Wolfe conditions
4: calculate the gradient $\nabla f(x_k)$
5: $x_{k+1} = x_k - \lambda_k \nabla f(x_k)$
6: $k = k + 1$

D

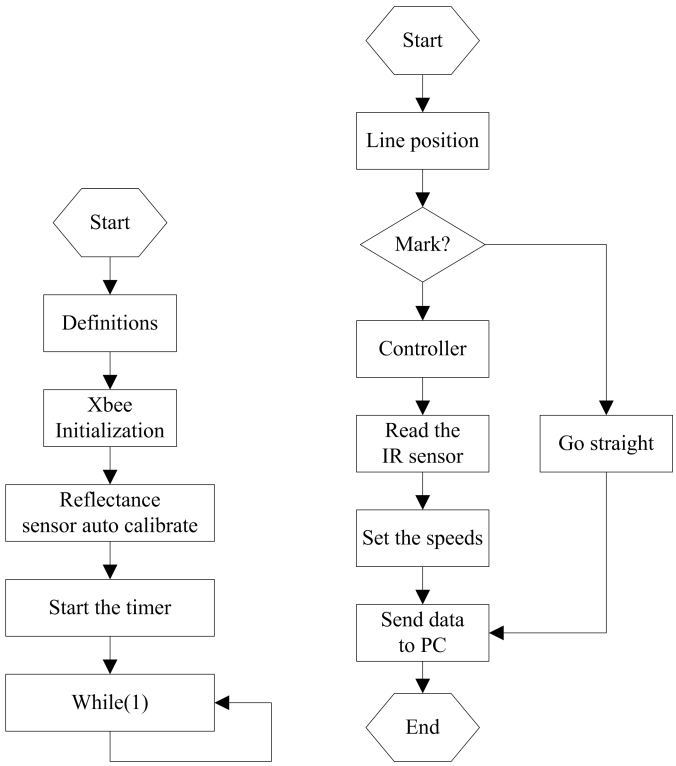
Experimental program

D.1 The mbed program

The flowchart of the mbed program is shown as Figure D.1. The time intervals of the robot' inputs are required for the localization. In order to simplify the work, the main code is in the timer interrupt function. This function is called repeatedly at a specified rate. The time intervals are always the same as that the timer overflows. The Ticker interface provided in the mbed SDK can be used to set up the recurring interrupt. The time interval must be greater than the maximum time the timer interrupt function runs once. In our case, the time interval is set to 0.1 s.

Two 12-bit ADCs are used to obtain the voltages of the IR sensors. In our experiments, they are p17 and p19 shown in Figure 6.2. The return value of ADC function is a 12-bit integer. The integer range [0, 0xFFF] corresponds to the voltage range 0 to 3.3 V.

The robot can follow the track with a simple controller. If the robot does not deviate too much, it will go straight. If not, it will turn to the opposite side. If the mark is detected by the robot and the feedback will be that the robot deviates too much. However, it is not that case. Before the robot meets the mark, the



(a) Main function (b) Timer interrupt function

Figure D.1: Flowchart of the mbed program

simple controller can ensure the robot does not deviate too much. At the mark, going straight is better than turning to one side. In order to obtain enough data for mapping problem, the robot should run slowly. The 3pi robot library for the mbed is provided to send the commands to the 3pi robot and receive the feedback returned by the 3pi robot. The return value of reflectance sensors is transformed from [0, 4000] into [-1.0, 1.0]. The range of the parameter for the speed is also transformed from [-255, 255] into [-1.0, 1.0].

For the XBee module connected to the mbed, the first step is to configure the definitions RADIO_TX, RADIO_RX and RADIO_RESET in config.h. They are assigned by the expansion board. The corresponding pins are p28, p27 and p26. The second step is to initial the XBee with these pins and baud rate which is the default setting (9600). Before sending data to PC, the MAC address of the XBee module connected to PC is required, which can be checked by XCTU. The data sent to PC is shown as Figure D.2. The details are described as follows:

Byte	0	1	2	3	4	5	6	7
	DN	HS1	LS1	HS2	LS2	LSP	RSP	MN

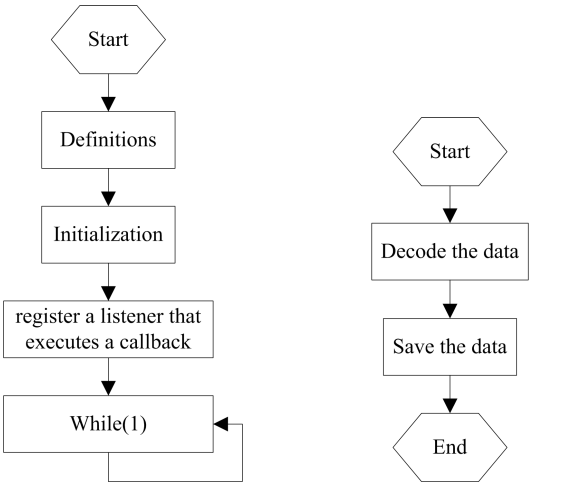
Figure D.2: Byte assignments of the data

- Byte 0 (DN): DN is the data number, in case the data is not received by PC in order. It is a number from 0 to 125. It will increase 1 when the data is sent once. When it becomes 126, it is changed to 0.

- Byte 1 - 4 (HS1, LS1, HS2, and LS2): The return value of the ADC is 12-bit. The 12-bit number is divided into two 6-bit numbers: H- and L-. The sampling voltage of IR sensor 1 consists of HS1 and LS1. The sampling voltage of IR sensor 2 consists of HS2 and LS2.
- Byte 5 - 6 (LSP and RSP): LSP and RSP are the speeds of the left wheel and the right wheel, respectively. The values are from 0 to 127.
- Byte 7 (MN), MN is used to indicate if the mark is detected at this moment.

D.2 The PC program

The flowchart of the PC program is shown as Figure D.3. It also consists of two parts: the main function and the callback function. In the main function, the XBee 2 is initialized with the port number and baud rate (9600). A listener is registered to execute the callback function when new data is received. In the callback function, the main task is to decode and save the message sent by the robot. The data is sent by the robot every 0.1s. As a result, the callback function will be executed with the same frequency. In case that the data cannot receive by PC in time, less code should be in the callback function. Except the decoding, the other tasks can be put in the main function.



(a) Main function (b) Callback function

Figure D.3: Flowchart of the PC program

Bibliography

[AMHHS17] Ali-Akbar Agha-Mohammadi, Eric Heiden, Karol Hausman, and Gaurav Sukhatme. Confidence-rich grid mapping. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, pages 1–19, 2017.

[ATT08] AliAkbar Aghamohammadi, Amir H Tamjidi, and Hamid D Taghirad. Slam using single laser range finder. *IFAC Proceedings Volumes*, 41(2):14657–14662, 2008.

[ATTM07] Ali Akbar Aghamohammadi, Hamid D Taghirad, Amir Hossein Tamjidi, and Ehsan Mihankhah. Feature-based laser scan matching for accurate and high speed mobile robot localization. In *Proceedings of the 3rd European Conference on Mobile Robots*, pages 1–6, 2007.

[BF95] Yoshua Bengio and Paolo Frasconi. An input output hmm architecture. In *Advances in neural information processing systems*, pages 427–434, 1995.

[BH09] Gabriele Bleser and Gustaf Hendeby. Using optical flow as lightweight slam alternative. In *Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality*, pages 175–176. IEEE, 2009.

[BH10] Mohamed Essayed Bouzouraa and Ulrich Hofmann. Fusion of occupancy grid mapping and model based object tracking for driver assistance systems using laser and radar sensors. In *2010 IEEE Intelligent Vehicles Symposium (IV)*, pages 294–300. IEEE, 2010.

[Bil08] Patrick Billingsley. *Probability and measure*. John Wiley & Sons, 2008.

[BLST02] Rahul Biswas, Benson Limketkai, Scott Sanner, and Sebastian Thrun. Towards object mapping in non-stationary environments with mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 1014–1019. IEEE, 2002.

[BM92] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992.

[BS03] Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, volume 3, pages 2743–2748. IEEE, 2003.

[Cho96] Howie Choset. *Sensor based motion planning: The hierarchical generalized Voronoi graph*. PhD thesis, California Institute of Technology, USA, 1996.

[CLLH07] H Jacky Chang, CS George Lee, Yung-Hsiang Lu, and Y Charlie Hu. P-slam: Simultaneous localization and mapping with environmental-structure prediction. *IEEE Transactions on Robotics*, 23(2):281–293, 2007.

[CN06] David M Cole and Paul M Newman. Using laser range data for 3d slam in outdoor environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1556–1563. IEEE, 2006.

[CPL+06] Christophe Coué, Cédric Pradalier, Christian Laugier, Thierry Fraichard, and Pierre Bessière. Bayesian occupancy filtering for multitarget tracking: an automotive application. *The International Journal of Robotics Research*, 25(1):19–30, 2006.

[CTLM06] Cheng Chen, Christopher Tay, Christian Laugier, and Kamel Mekhnacha. Dynamic environment modeling with gridmap: a multiple-object tracking application. In *Proceedings of the 9th IEEE International Conference on Control, Automation, Robotics and Vision*, pages 1–6. IEEE, 2006.

[DDK15] Nitin Kumar Dhiman, Dipti Deodhare, and Deepak Khemani. Where am i? creating spatial awareness in unmanned ground robots using slam: A survey. *Sadhana*, 40(5):1385–1433, 2015.

[DHS+14] Renaud Dubé, Markus Hahn, Markus Schutz, Jurgen Dickmann, and Denis Gingras. Detection of parked vehicles from a radar based occupancy grid. In *Proceedings of the IEEE International Conference on Intelligent Vehicles Symposium Proceedings*, pages 1415–1420. IEEE, 2014.

[DK04] Albert Diosi and Lindsay Kleeman. Advanced sonar and laser range finder fusion for simultaneous localization and mapping. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, volume 2, pages 1854–1859. IEEE, 2004.

[DKS15] Abhinav Dadhich, Nishanth Koganti, and Tomohiro Shibata. Modeling occupancy grids using edhmm for dynamic environments. In *Proceedings of the 2015 Conference on Advances In Robotics*, pages 60:1–60:6. ACM, 2015.

[DNDW+00] MWM Gamini Dissanayake, Paul Newman, Hugh F Durrant-Whyte, Steve Clark, and M Csorba. An experimental and theoretical investigation into simultaneous localisation and map building. In *Experimental robotics VI*, pages 265–274. Springer, 2000.

[DON11] Radu Danescu, Florin Oniga, and Sergiu Nedevschi. Modeling and tracking the driving environment with a particle-based occupancy grid. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1331–1342, 2011.

[DRMS07] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[DSB17] David Droeschel, Max Schwarz, and Sven Behnke. Continuous mapping and localization for autonomous navigation in rough terrain using a 3d laser scanner. *Robotics and Autonomous Systems*, 88:104–115, 2017.

[DWB06] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping (slam): Part i. *IEEE Robotics and Automation Magazine*, 13(2):99–110, 2006.

[DWE16] Kevin Doherty, Jinkun Wang, and Brendan Englot. Probabilistic map fusion for fast, incremental occupancy mapping with 3d hilbert maps. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1011–1018. IEEE, 2016.

[DWW12] Michael Dewar, Chris Wiggins, and Frank Wood. Inference in hidden markov models with explicit state duration distributions. *IEEE Signal Processing Letters*, 19(4):235–238, 2012.

[Elf86] Alberto Elfes. A sonar-based mapping and navigation system. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 1151–1156. IEEE, 1986.

[Elf89] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[ESG10] Erik Einhorn, Christof Schröter, and Horst-Michael Gross. Building 2d and 3d adaptiveresolution occupancy maps using nd-trees. *Proceedings of the 55th International Scientific Colloquiium*, pages 306–311, 2010.

[FDvNV10] Okke Formsma, Nick Dijkshoorn, Sander van Noort, and Arnoud Visser. Realistic simulation of laser range finder behavior in a smoky environment. In *Proceedings of the Robot Soccer World Cup*, pages 336–349. Springer, 2010.

[GAVA08] Ruben Garcia, Olivier Aycard, Trung-Dung Vu, and Malte Ahrholdt. High level sensor data fusion for automotive applications using occupancy grids. In *Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision*, pages 530–535. IEEE, 2008.

[GLGMM+11] Ernesto Martín Gorostiza, José Luis Lázaro Galilea, Franciso Javier Meca Meca, David Salido Monzú, Felipe Espinosa Zapata, and Luis Pallarés Puerto. Infrared sensor system for mobile-robot positioning in intelligent spaces. *Sensors*, 11(5):5416–5438, 2011.

[GNB00] Jose Guivant, Eduardo Nebot, and Stephan Baiker. Autonomous navigation and map building using laser range sensors in outdoor applications. *Journal of robotic systems*, 17(10):565–583, 2000.

[GSNR11] Karl Granström, Thomas B Schön, Juan I Nieto, and Fabio T Ramos. Learning to close loops from range data. *The international journal of robotics research*, 30(14):1728–1754, 2011.

[HDLF19] Jiawei Huang, Mahmut Demir, Thang Lian, and Kikuo Fujimura. An online multi-lidar dynamic occupancy mapping method. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 517–522. IEEE, 2019.

[HK01] Andrew Heale and Lindsay Kleeman. Fast target classification using sonar. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, volume 3, pages 1446–1451. IEEE, 2001.

[HKD07] HD Herath, S Kodagoda, and G Dissanayake. *Stereo vision based SLAM: Issues and solutions*. ITECH, 2007.

[HN06] Kin Leong Ho and Paul Newman. Loop closure detection in slam by combining visual and spatial appearance. *Robotics and Autonomous Systems*, 54(9):740–749, 2006.

[HR16] Markus Hammarsten and Viktor Runemalm. 3d localization and mapping using automotive radar. Master's thesis, Chalmers University of Technology, Sweden, 2016.

[JKK16] Ruben Jungnickel, Michael Köhler, and Franz Korf. Efficient automotive grid maps using a sensor ray based refinement process. In *Proceedings of the IEEE International Conference on Intelligent Vehicles Symposium (IV)*, pages 668–675. IEEE, 2016.

[Jul02] Simon J Julier. The scaled unscented transformation. In *Proceedings of the American Control Conference*, volume 6, pages 4555–4559. IEEE, 2002.

[KB91] Benjamin Kuipers and Yung-Tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and autonomous systems*, 8(1-2):47–63, 1991.

[KC16] Jiwoong Kim and Woojin Chung. Localization of a mobile robot using a laser range finder in a glass-walled environment. *IEEE Transactions on Industrial Electronics*, 63(6):3616–3627, 2016.

[KGU04] Gerhard K Kraetzschmar, Guillem Pages Gassull, and Klaus Uhl. Probabilistic quadtrees for variable-resolution mapping of large environments. *IFAC Proceedings Volumes*, 37(8):675–680, 2004.

[KK95] Lindsay Kleeman and Roman Kuc. Mobile robot sonar for target localization and classification. *The International Journal of Robotics Research*, 14(4):295–318, 1995.

[KK11] Soohwan Kim and Jonghyuk Kim. Towards large-scale occupancy map building using dirichlet and gaussian processes. In *Proceedings of the Australasian Conference on Robotics and Automation*, pages 4756–4761, 2011.

[KK12] Soohwan Kim and Jonghyuk Kim. Building occupancy maps with a mixture of gaussian processes. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4756–4761. IEEE, 2012.

[KK13] Soohwan Kim and Jonghyuk Kim. Continuous occupancy maps using overlapping local gaussian processes. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4709–4714. IEEE, 2013.

[KKCR18] Timo Korthals, Mikkel Kragh, Peter Christiansen, and Ulrich Rückert. Towards inverse sensor mapping in agriculture. *arXiv preprint arXiv:1805.08595*, 2018.

[KLAM16] Evan Kaufman, Taeyoung Lee, Zhuming Ai, and Ira S Moskowitz. Bayesian occupancy grid mapping via an exact inverse sensor model. In *American Control Conference (ACC)*, pages 5709–5715. IEEE, 2016.

[KMEM11] Kurt Konolige, Eitan Marder-Eppstein, and Bhaskara Marthi. Navigation in hybrid metric-topological maps. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3041–3047. IEEE, 2011.

[KTR+17] Daisuke Kakuma, Satoki Tsuichihara, Gustavo Alfonso Garcia Ricardez, Jun Takamatsu, and Tsukasa Ogasawara. Alignment of occupancy grid and floor maps using graph matching. In *Proceedings of the 11th International Conference on Semantic Computing (ICSC)*, pages 57–60. IEEE, 2017.

[LBaR] Hongjun Li, Miguel Barão, and Luís Rato. Online learning occupancy grid maps for mobile robots. In *2nd International Workshop on Sustainability and Green Technologies*, pages 1–13, 2017.

[LBaR18a] Hongjun Li, Miguel Barão, and Luís Rato. Gaussian random field-based log odds occupancy mapping. In *Proceedings of IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, pages 1–4. IEEE, 2018.

[LBaR18b] Hongjun Li, Miguel Barão, and Luís Rato. Mapping dynamic environments using markov random field models. In *Proceedings of 24th International Conference on Automation and Computing (ICAC)*, pages 1–5, 2018.

[LBaR18c] Hongjun Li, Miguel Barão, and Luís Rato. Markov random field-based prediction for mobile robots. In *Eighth Workshop in Informatics of University of Evora*, pages 1–8, 2018.

[LC11] Geunho Lee and Nak Young Chong. Low-cost dual rotating infrared sensor for mobile robot swarm applications. *IEEE Transactions on Industrial Informatics*, 7(2):277–286, 2011.

[LDW91] John J Leonard and Hugh F Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings of the International Workshop on Intelligent Robots and Systems*, pages 1442–1447. Ieee, 1991.

[LH02] Joo-Ho Lee and Hideki Hashimoto. Intelligent space—concept and contents. *Advanced Robotics*, 16(3):265–280, 2002.

[Li09] Stan Z Li. *Markov random field modeling in image analysis*. Springer Science & Business Media, 2009.

[LM97] Feng Lu and Evangelos Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic systems*, 18(3):249–275, 1997.

[Low99] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. IEEE, 1999.

[Low04] Kok-Lim Low. Linear least-squares optimization for point-to-plane icp surface registration. *Chapel Hill, University of North Carolina*, 4(10):1–3, 2004.

[LZHL19] Bhoram Lee, Clark Zhang, Zonghao Huang, and Daniel D Lee. Online continuous mapping using gaussian process implicit surfaces. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6884–6890. IEEE, 2019.

[MD08] Gianluigi Mongillo and Sophie Deneve. Online learning with hidden markov models. *Neural computation*, 20(7):1706–1716, 2008.

[MDBB12] Daniel Meyer-Delius, Maximilian Beinhofer, and Wolfram Burgard. Occupancy grid models for robot mapping in changing environments. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 2024–2030, 2012.

[ME85] Hans Moravec and Alberto Elfes. High resolution maps from wide angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 116–121. IEEE, 1985.

[ML11] Naveed Muhammad and Simon Lacroix. Loop closure detection using small-sized signatures from 3d lidar data. In *Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 333–338. IEEE, 2011.

[MLM05] Javier Minguez, Florent Lamiraux, and Luis Montesano. Metric-based scan matching algorithms for mobile robot displacement estimation. In *ICRA*, pages 3557–3563, 2005.

[MMRL08] Kamel Mekhnacha, Yong Mao, David Raulo, and Christian Laugier. Bayesian occupancy filter based" fast clustering-tracking" algorithm. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 1–8, 2008.

[MS04] James McLurkin and Jennifer Smith. Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems*, pages 1–10. Citeseer, 2004.

[MSDB03] M Michael, T Sebastian, K Daphne, and W Ben. Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1–6, 2003.

[MT07] Nikos C Mitsou and Costas S Tzafestas. Temporal occupancy grid for mobile robot dynamic environment mapping. In *Proceedings of the Mediterranean Conference on Control & Automation*, pages 1–8. IEEE, 2007.

[MTKW02] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593–598, 2002.

[NCH06] Paul Newman, David Cole, and Kin Ho. Outdoor slam using visual appearance and laser ranging. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1180–1187. IEEE, 2006.

[NRL14] Amaury Nègre, Lukas Rummelhard, and Christian Laugier. Hybrid sampling bayesian occupancy filter. In *Proceedings of the IEEE International Conference on Intelligent Vehicles Symposium*, pages 1307–1312. IEEE, 2014.

[NYK+15] Dominik Nuss, Ting Yuan, Gunther Krehl, Manuel Stuebler, Stephan Reuter, and Klaus Dietmayer. Fusion of laser and radar sensor data with a sequential monte carlo bayesian occupancy filter. In *Proceedings of the IEEE International Conference on Intelligent Vehicles Symposium (IV)*, pages 1074–1081. IEEE, 2015.

[OR12] Simon T O'Callaghan and Fabio T Ramos. Gaussian process occupancy maps. *The International Journal of Robotics Research*, 31(1):42–62, 2012.

[OTM18] Cormac O'Meadhra, Wennie Tabib, and Nathan Michael. Variable resolution occupancy mapping using gaussian mixture models. *IEEE Robotics and Automation Letters*, 4(2):2015–2022, 2018.

[PKK93] Nikolaos P Papanikolopoulos, Pradeep K Khosla, and Takeo Kanade. Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision. *IEEE transactions on robotics and automation*, 9(1):14–35, 1993.

[PKRB02] Sam T Pfister, Kristo L Kriechbaum, Stergios I Roumeliotis, and Joel W Burdick. Weighted range sensor matching algorithms for mobile robot displacement estimation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1667–1674. IEEE, 2002.

[Rab89] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[Rat02] Luís Rato. *Controlo Comutado Baseado em Modelos Múltiplos*. PhD thesis, Technical University of Lisbon, Portugal, 2002.

[RDH+16] Matthias Rapp, Klaus Dietmayer, Markus Hahn, Bharanidhar Duraisamy, and Jürgen Dickmann. Hidden markov model-based occupancy grid maps of dynamic environments. In *Proceedings of the 19th International Conference on Information Fusion (FUSION)*, pages 1780–1788. IEEE, 2016.

[RM09] Julien Ros and Kamel Mekhnacha. Multi-sensor human tracking with the bayesian occupancy filter. In *Proceedings of the 16th International Conference on Digital Signal Processing*, pages 1–8. IEEE, 2009.

[RO16] Fabio Ramos and Lionel Ott. Hilbert maps: scalable continuous occupancy mapping with stochastic gradient descent. *The International Journal of Robotics Research*, 35(14):1717–1730, 2016.

[RRN10] David Ribas, Pere Ridao, and José Neira. *Underwater SLAM for structured environments using an imaging sonar*, volume 65. Springer, 2010.

[RSZF09] James F Roberts, Timothy S Stirling, Jean-Christophe Zufferey, and Dario Floreano. 2.5 d infrared range and bearing system for collective robotics. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 3659–3664. IEEE, 2009.

[RW06] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian Processes for Machine Learning*. London: MIT Press, 2006.

[SADD14] Markus Schutz, Nils Appenrodt, Jurgen Dickmann, and Klaus Dietmayer. Occupancy grid map-based extended object tracking. In *Proceedings of the IEEE International Conference on Intelligent Vehicles Symposium Proceedings*, pages 1205–1210. IEEE, 2014.

[SD19] Michael Slutsky and Daniel Dobkin. Dual inverse sensor model for radar occupancy grids. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1760–1767. IEEE, 2019.

[SHBG05] Cyrill Stachniss, Dirk Hähnel, Wolfram Burgard, and Giorgio Grisetti. On actively closing loops in grid-based fastslam. *Advanced Robotics*, 19(10):1059–1079, 2005.

[SK04] Francesco Savelli and Benjamin Kuipers. Loop-closing and planarity in topological map-building. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, volume 2, pages 1511–1517. IEEE, 2004.

[SNKB07] Daejung Shin, Seung You Na, Jin Young Kim, and Seong-Joon Baek. Sonar localization using ubiquitous sensor network for water pollution monitoring fish robots. In *Proceedings of the IEEE International Symposium on Signal Processing and Information Technology*, pages 80–85. IEEE, 2007.

[SNS15] Daniel Perea Ström, Fabrizio Nenci, and Cyrill Stachniss. Predictive exploration considering previously mapped environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2761–2766. IEEE, 2015.

[SS01] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[SSC90] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. London: MIT press, 2005.

[Thr01] Sebastian Thrun. Learning occupancy grids with forward models. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1676–1681. IEEE, 2001.

[TPB06] Rudolph Triebel, Patrick Pfaff, and Wolfram Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 2276–2282. IEEE, 2006.

[Tre00] Volker Tresp. A bayesian committee machine. *Neural computation*, 12(11):2719–2741, 2000.

[TTW+04] Sebastian Thrun, Scott Thayer, William Whittaker, Christopher Baker, Wolfram Burgard, David Ferguson, Dirk Hahnel, D Montemerlo, Aaron Morris, Zachary Omohundro, et al. Autonomous exploration and mapping of abandoned mines. *IEEE Robotics & Automation Magazine*, 11(4):79–91, 2004.

[TTWB14] Georg Tanzmeister, Julian Thomas, Dirk Wollherr, and Martin Buss. Grid-based mapping and tracking in dynamic environments using a uniform evidential environment representation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 6090–6095. IEEE, 2014.

[Tu15] Stephen Tu. Derivation of baum-welch algorithm for hidden markov models, 2015.

[VDM04] Rudolph Van Der Merwe. *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. PhD thesis, Oregon Health & Science University, USA, 2004.

[VR16] Carlos EO Vido and Fabio Ramos. From grids to continuous occupancy maps through area kernels. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1043–1048. IEEE, 2016.

[WAJF14] Zhan Wang, Rares Ambrus, Patric Jensfelt, and John Folkesson. Modeling motion patterns of dynamic objects by iohmm. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1832–1838. IEEE, 2014.

[WE16] Jinkun Wang and Brendan Englot. Fast, accurate gaussian process occupancy maps via test-data octrees and nested bayesian fusion. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1003–1010. IEEE, 2016.

[WL10] Xiao-hua Wang and Peng-fei Li. Improved data association method in binocular vision-slam. In *Proceedings of the 2010 International Conference on Intelligent Computation Technology and Automation*, volume 2, pages 502–505. IEEE, 2010.

[WLH+17] Timo Winterling, Jakob Lombacher, Markus Hahn, Jürgen Dickmann, and Christian Wöhler. Optimizing labelling on radar-based grid maps using active learning. In *Proceedings of the IEEE International Conference on Radar Symposium*, pages 1–6. IEEE, 2017.

[WN99] Stephen Wright and Jorge Nocedal. Numerical optimization. *Springer Science*, 35(67–68):7, 1999.

[WS04] Denis Wolf and Gaurav S Sukhatme. Online simultaneous localization and mapping in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1301–1307. IEEE, 2004.

[WT02] Chieh-Chih Wang and Chuck Thorpe. Simultaneous localization and mapping with detection and tracking of moving objects. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 2918–2924. IEEE, 2002.

[WTT+07] Chieh-Chih Wang, Charles Thorpe, Sebastian Thrun, Martial Hebert, and Hugh Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research*, 26(9):889–916, 2007.

[WU18] Joseph Wessner and Wolfgang Utschick. Extending occupancy grid mapping for dynamic environments. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 701–707. IEEE, 2018.

[WVDM00] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158. Ieee, 2000.

[WVW12] Emanuel Walldén Viklund and Johan Wågberg. Continuous occupancy mapping using gaussian processes. Master's thesis, Linköping University, Sweden, 2012.

[Yu10] Shun-Zheng Yu. Hidden semi-markov models. *Artificial intelligence*, 174(2):215–243, 2010.

UNIVERSIDADE DE ÉVORA
INSTITUTO DE INVESTIGAÇÃO
E FORMAÇÃO AVANÇADA

Contactos:
Universidade de Évora
Instituto de Investigação e Formação Avançada — IIFA
Palácio do Vimioso | Largo Marquês de Marialva, Apart. 94
7002 - 554 Évora | Portugal
Tel: (+351) 266 706 581
Fax: (+351) 266 744 677
email: iifa@uevora.pt